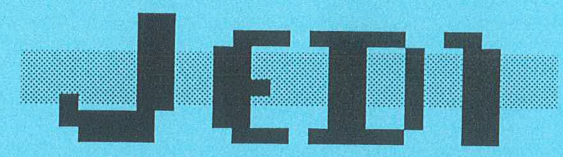
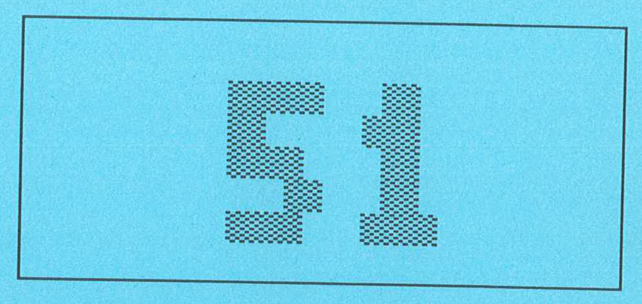




JUIN 1989

LE JOURNAL QUI NE S'ENDORT PAS EN SURSAUT

LA VIRGULE FLOTTANTE CORDIC

LE FORUM TELEMATIQUE SAM*JEDI

EDITORIAL

Au vu de la présentation de ce sommaire, vous avez certainement constaté un changement de présentation. Nous sommes dorénavant en mesure de tirer maintenant les maquettes de la revue sur imprimante laser. Un scanner à main permet aussi l'introduction de graphismes dans nos pages.

Concernant notre retard de parution, nous faisons une fois de plus appel aux bonnes volontés pour nous aider à remplir les pages de la revue.

Nous avons réussi enfin à établir un contact avec un club anglais; espérons que de fructueux échanges en découleront. A ceux qui partent en vacance, nous leur souhaitons du beau temps et de disposer d'assez de temps pour nous mijoter quelques programmes bourrés d'astuces. Puis-je compter sur vous? Ne me décevez pas.

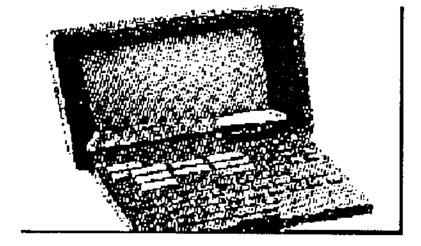
SOMMAIRE

Toute reproduction, adaptation, traduction partielle du contenu de ce magazine sous toutes les formes est vivement encouragée, à l'exception de toute reproduction à des fins commerciales. Dans le cas de reproduction par photocopie, il est demandé de ne pas masquer les références inscrites en bas ed page, et dans les autres cas, de citer l'ASSOCIATION JEDI (loi 1901).

Nos coordonnées: ASSOCIATION JEDI
17, rue de la Lancette
75012 PARIS
tél président: (33) 1-43.40.96.53
tél secrétaire: (33) 1-49.85.63.67

TELETEL:
3615 SAM*JEDI
(France, DOM, TOM)

(33) 36.43.15.15 SAM*JEDI
(1200,E,7,1) (Etranger)



PACKAGE VIRGULE FLOTTANTE
ALGORITHME DE CORDIC

par Christophe LAVARENNE

Système: TURBO-Forth 83-Standard
Adaptabilité: F83-Laxen et Perry MSDOS et CP/M
VolksFORTH ATARI
Diffusion: téléchargement 3615 SAM*JEDI "Maths"
et prochainement module M8

Pour la première fois dans les colonnes de JEDI, un programme TURBO-Forth est d'abord écrit et diffusé en anglais. De précédents numéros de JEDI (20, 38, 42 et 47) traitaient la virgule flottante en pratique et en théorie.

Cette nouvelle mouture devrait retenir votre attention, car ses caractéristiques en font un des programmes de traitement numérique les plus performants diffusés à ce jour dans nos colonnes:

- contrôle de la précision de calcul à la compilation (10 à plus de 30 décimales),
- gestion d'une pile de nombres flottants à part des piles données et retour,
- opérateurs arithmétiques, fonctions logarithme, exponentielles, trigonométriques,
- conversion polaire-scalaire,
- calculs sur nombres complexes.

Le programme ne comprend pas moins de huit fichiers. Selon la rapidité souhaitée, on peut compiler les routines d'optimisation écrites en assembleur FORTH 8086 (TURBO-Forth et F83 Laxen et Perry MSDOS) ou leur version FORTH sur d'autres systèmes.

LISTING: CORDICFP.FTH

```
\ ************************************************
\ TurboForth CORDIC Floating Point package v1.0
\
\ Makefile : see compiling options after EOF
\
\ Original code written by:
\ Christophe Lavarenne, consultant
\ 12, rue du Docteur Vuillieme
\ 92190 Meudon FRANCE       tel (+33)(1)45071341
\ ***********************************************

  cr .( Loading CORDIC Floating-Point package v1.0...) cr
decimal only forth also definitions
vocabulary CORDIC CORDIC also definitions


  include FPI24          \ (1)  <---------\ choose either (1)
\ 53 constant SMB        \ (2a) <---------| or (2a)+(2b)+(2c)
\ include FPIXX1         \ (2b) <---------|
                         \                |
\ include FP_FORTH       \ (A) \ choose   |
  include FP_8086        \ (B) / either   |
                         \                |
  include FPKERNEL       \ choose always  |
                         \                |
\ include FPIXX2         \ (2c) <---------/

   SMB dup s>f 2 s>f flog f*  \ -- SMB | == SMB*log(2)
   .(   accuracy: ) . .( bits =) 2 places f.
   .( decimal digits)

\ initial settings :
   6 PLACES   NEAREST   FLOATS   DEGREES    cr
 .( Floating-Point package loaded in vocabulary CORDIC) cr

  LEARN                 \ link CORDIC.VOC documentation
EOF
**************************************** COMPILING OPTIONS

Option on SMB, the number of Significant Mantissa Bits:
- for SMB=24, CORDIC constants are precomputed,
  choose (1) (choose this option for meta-compilation)
- for other values of SMB, CORDIC constants must be
initialized:
  choose (2a) and (2b) and (2c) and change SMB's value
  WARNING: SMB must NOT be less than 10
           the package needs about (n*n/11 + 11*n + 7,5k)
           bytes, with n = SMB + log2(SMB)

Option on the primitive definitions :
- choose (A) for high level Forth83 definitions
(portability)
- choose (B) for 8086 assembler definitions (10 times
faster)
```

LISTING: FPI24.FTH

```
\ ************************************************
\ TurboForth CORDIC Floating Point package v1.0
\
\ CORDIC constants initialization for SMB=24
\ (see CORDICFP.FTH for file compilation order)
\ ****************************************
\ constants     (24 bits fixed accuracy)

decimal
 24 constant SMB    \ number of Significant Mantissa Bits
 29 constant gw
    \ mantissa Guarded bit Width = SMB+log2(SMB)
  4 constant rw
    \ Register byte Width = (GW+3 +7)/8

\ ****************************************
\ registers

create x rw allot         \ x -= mS y>>i
create y rw allot         \ y +=  S x>>i
create z rw allot         \ z -=  S z[i]

create xs rw allot        \ secondary registers
create ys rw allot
create zs rw allot

HEX
\ ****************************************
\ linear tables (24 bits fixed accuracy)

create zlrw      \ cyclic table for negative powers of 2
  8000 , 0 ,  4000 , 0 ,  2000 , 0 ,  1000 , 0 ,
  0800 , 0 ,  0400 , 0 ,  0200 , 0 ,  0100 , 0 ,
  0080 , 0 ,  0040 , 0 ,  0020 , 0 ,  0010 , 0 ,
  0008 , 0 ,  0004 , 0 ,  0002 , 0 ,  0001 , 0 ,  0 , 0 ,

create zl2   zlrw  dup 3 rw * 2- +  \ -- zlrw zl[1]
  dup , rw + dup , rw + dup , rw + dup , rw +
  dup , rw + dup , rw + dup , rw + dup , rw +
  dup , rw + dup , rw + dup , rw + dup , rw + ,
  dup , rw + dup , rw + dup , rw + dup , rw +
  dup , rw + dup , rw + dup , rw + dup , rw +
  dup , rw + dup , rw + dup , rw + dup , rw +
  dup , rw + dup , rw + dup , rw + ,

\ ****************************************************
\ trigo constants and tables  (24 bits fixed accuracy)

create 1/Kt  9DB6 , 136E ,    \ Trigo radius scaling factor
create r>d   DC1A , 1CA5 ,    \ 180/pi>>6
create d>r   46A3 , 11DF ,    \ pi/180>>-5
create r>g   BBB2 , 1FD4 ,    \ 200/pi>>6
create g>r   BF93 , 1015 ,    \ pi/200>>-5

create ztrw
   \ Atg(1>>i) i=0..gw/3. i>gw/3 => Atg(1>>i)=1>>i
  FB55 , 1921 , 3384 , 0ED6 , DD7E , 07D6 , B753 , 03FA ,
  55BC , 01FF , EAAE , 00FF , FD56 , 007F , FFAB , 003F ,
  FFF6 , 001F ,  FFFF , 000F ,

create zt2      ztrw
  dup ,, rw + dup , rw + dup , rw + dup , rw + dup , rw +
  dup , rw + dup , rw + dup , rw + dup , rw + ,
zl2 gw 3 / 2* +  here  gw dup 3 / - 2*  dup allot  cmove

create zt1     \ for XYROT
  1 c, 2 c, 3 c, 3 c, 4 c, 2 c, 3 c, 2 c, 4 c, 2 c,
  3 c, 1 c, 1 c, 1 c, 1 c, 1 c, 1 c, 1 c, 1 c, 1 c,
  1 c, 1 c, 1 c, 1 c, 1 c, 1 c, 1 c, 1 c, 1 c, 1 c,

\ ****************************************************
\ hyper constants and tables  (24 bits fixed accuracy)
```

```
create 1/Kh     D0FE , 26A3 ,
   \ Hyper radius scaling factor
create ln2       42F4 , 162E ,    \ ln(2)
create ln10/4   B1B4 , 126B ,    \ ln(10)>>2

create zhrw
   \ Ath(1>>i) i=1..gw/3. i>gw/3 => Ath(1>>i)=1>>i
  EA76 , 1193 , 577A , 082C , 6245 , 0405 , AB10 , 0200 ,
  1558 , 0100 , 02AA , 0080 , 0055 , 0040 , 000A , 0020 ,
  0001 , 0010 ,

create zh2  zhrw
  dup , rw + dup , rw + dup , rw + dup , rw + dup , rw +
  dup , rw + dup , rw + dup , rw + ,
zl2 gw 3 / 2* +  here  gw dup 3 / - 2*  dup allot  cmove

DECIMAL
EOF
/ *****************************************

: .table        \ ad i --                  \ dump table
  cr over body> >name .name                \ table name
  ." : key to dump/stop..." key drop    \
  gw 1+ swap                             \ for each table item
  do   key? if key key 2drop then         \ stop if key
       cr i 2 .r  dup @                    \ print item index
       dup h.  3 spaces                  \ print mantissa in hex
       z swap :=  0 z reg@  e.   \ print item in E format
  2+ loop drop ;

zl2 1 .table                  \ dump linear table
zh2 1 .table                  \ dump hyper  table
zt2 0 .table                  \ dump trigo  table
forget .table

EOF
*****************************************
```

This code initializes statically the Cordic tables (instead of computing them at compile time) so that the file may easily be cross-commpiled for a number of Significant Mantissa Bits fixed to SMB=24

Each register or value is a mantissa, rw bytes wide, with byte significance increasing as byte address, in 2's complement in the range [-4.0,4.0[(fix point left of msb-3 : highest byte of 0.5 is hex 10)

Cross compilation for other values of SMB would ask to generate the same kind of file, where the constant values would be copied from the tables dump (see .table)

LISTING: FPIXX1.FTH

```
\ ********************************************
\ TurboForth CORDIC Floating Point package v1.0
\
\ CORDIC initialization for variable accuracy (part1)
\ (see CORDICFP.FTH for files compilation order)
\ ********************************************


SMB 10 < abort" SMB should be 10 minimum"
              \ else rw<4 can't hold double integers

: gw      \ n -- n+log2(n)            \ add guard bits
  dup                                 \ for i=0 to log2(SMB)
  begin  1+  swap 2/ tuck  0=  until   \   n>>i n+i --
  nip ;                               \ -- n+log2(n)

SMB gw   forget gw     constant gw   \ Guarded bit Width
gw  3 +  15 + 16 / 2*   constant rw   \ Register byte Width

\ z?rw tables hold precomputed values (rw bytes wide)
\ z?2  tables hold values address for each step
\ (2 bytes wide)

create zlrw   17 rw * allot     \ negative powers of 2
create zl2    gw 2* allot     \ *zl2[i]=1>>i       i=1..gw

create zhrw   gw 3 / rw * allot
   \ zhrw[i]=Ath(1>>i) i=1..gw/3
create zh2    gw 2* allot
   \ *zh2[i]=Ath(1>>i) i=1..gw

create ztrw   gw 3 / 1+ rw * allot
   \ ztrw[i]=Atg(1>>i) i=0..gw/3
create zt2    gw 1+ 2* allot
   \ *zt2[i]=Atg(1>>i) i=0..gw
create zt1    gw 1+ allot
   \ trigo repeat bytes for Kt=2

create ln2     rw allot    \ natural logarithm of 2
create ln10/4  rw allot    \ 1/4 natural logarithm of 10
create 1/Kh    rw allot    \ hyper radius scaling factor

create 1/Kt    rw allot    \ trigo radius scaling factor
create r>d     rw allot    \ (180/pi)>>6
create d>r     rw allot    \ (pi/180)>>-5
create r>g     rw allot    \ (200/pi)>>6
create g>r     rw allot    \ (pi/200)>>-5

\ The Cordic algorithm uses 3 main registers
\ 3 secondary registers are used to hold
\ intermediate results

create x rw allot                  \ x -= mS y>>i
create y rw allot                  \ y +=  S x>>i
create z rw allot                  \ z -=  S z[i]

create xs rw allot
create ys rw allot
create zs rw allot

EOF

********************************************
```

Cordic initializations for variable accuracy

This code computes constants from SMB (the number of Significant Mantissa Bits).

SMB IS THE ONLY CONSTANT YOU HAVE TO CHANGE (SMB is defined in FPMAKE.FTH)

All following registers, tables, constants, and primitives are parametrized by SMB.

Tables and constants values are computed using primitives (see FPIXX2.FTH; primitives are defined in FP_xxx.FTH)

Mantissae hold values in 2's complement format; integer bit is at msb-2: most = +/-3.999... (-4.0 unnormable) least significant bit is at msb-3-SMB

LISTING: FPIXX2.FTH

```
\ ********************************************
\ TurboForth CORDIC Floating Point package v1.0
\
\ Constants initialization for sizable accuracy (part2)
\ (see CORDICFP.FTH for files compilation order)
\ ********************************************

create init                        \ tag to forget

: initzlrw      \ --               \ init. zlrw table
  z clear  1 z rw + 2- !           \ zlrw[00]=00008000...
  zlrw  17 1                       \ zlrw[01]=00004000...
  do   i over z >>=  rw +          \ zlrw[15]=00000001...
  loop clear ;                     \ zlrw[16]=all zero

: initzl2       \ --                  \ init zl2 table
  zl2  gw 0                           \
  do   i 3 + rw * 2-                  \
       [ 16 rw * 2- ] literal mod     \
       zlrw + over !  2+              \ zl2[i] = 1>>i
  loop drop ;                         \ --

: initzrw          \ compute Gatan(1>>i)
  zhrw gw 3 / rw *  ztrw over rw +  \ erase ztrw and zhrw
  erase erase  gw 2/ 0              \ for j=0 to gw/2-1
  do  x clear  i 2* 1+  x rw + 2- ! \ x=(2j+1)>>13
      13 x norm  y 2 /4=            \ y=1/2
      vectoring linear  2- z dup >>= \ z = 1/(2j+1) = N
      ztrw rw +  zhrw               \ -- ztrw[1] zhrw[1]
      gw  i 2* 1+  3 max  /  0      \ for i=1 to gw/(2j+1)
      do  j 2* 1+ z dup >>=         \ z=N>>[i*(2j+1)]
          dup z +=            rw + SWAP \ zhrw[i] += z
          dup z  j 1 and            \ ztrw[i] += z*(-1)^j
          if -= else += then  rw + SWAP
```

```
          \ -- ztrw[i+1] zhrw[i+1]
      loop 2drop
  loop ;
\ i<=gw/3: Gatan(1>>i)
\   = sum(j>=0):[(-m^j)/(2j+1)] >> [i(2j+1)]
\ i>gw/3: Gatan(1>>i) = 1>>i

: initz2        \ adrw ad2 n --   \ initialize either:
  gw 3 / 1+ swap                  \ zhrw zh2 1 initz2
  do   2dup !                     \ ztrw zt2 0 initz2
       swap rw + swap 2+
  loop nip
  zl2 gw 3 / 2* + swap
  gw dup 3 / - 2* cmove ;

: init1/Kh      \ --              \ hyper rescaling factor
  x 2 /4=  z clear  rotating hyper \ x=Kh/2 >1/4
  y 2 /4=  vectoring linear       \ z=1/2Kh
  z dup +=  1/Kh z := ;

: initln2       \ --              \ natural log of 2
  z 2 /4=  (lnsr)                 \ z=ln(1/2)/2
  z +/-  z dup +=                 \ z=-ln(1/2)=ln(2)
  ln2 z := ;

: initln10/4    \ --              \ 1/4 natural log of 10
  10 s>f fln ln10/4 reg! drop ;

: initpi/4      \ --              \ pi/4=Atan(1)
  x 4 /4=  y 4 /4=  vectoring     \ x=1 y=1 z=0
  zt2 2+  gw 1
  do   i (trigo)  2+
  loop drop                        \ --
  ztrw z :=  ;                     \ ztrw[0]=pi/4
\ the vectoring trigo cycle without i=0 has a domain of
\ convergence greater than pi/4, therefore it may be used
\ to compute pi/4 = Atg(1) = Atg(y/x) with x=y

: init1/Kt      \ --               \ trigo rescaling factor
  x 4 /4=  z clear  rotating trigo \ x=Kt<2
  y 4 /4=  vectoring linear        \ z=1/2Kt
  z dup +=  1/Kt z := ;

: initrad       \ --               \ for rad> and >rad
  180 s>f pi f/ r>d reg! drop
  pi 180 s>f f/ d>r reg! drop
  200 s>f pi f/ r>g reg! drop
  pi 200 s>f f/ g>r reg! drop ;

create kt rw allot                 \ kt-->1 => Kt-->2

: initzt1       \ --               \ trigo repeat bytes
  kt 2 /4=  gw 2/ 1                \ kt = [1+(1>>0)^2]/4
  do  i 2* x kt >>=  kt x +=       \ kt *= 1 + [1>>i]^2
  loop                             \ kt = (Kt^2)/4
  zt1  gw 2/ 0                     \ for i=0 to gw/2-1
  do 0 begin 1+                    \   -- zt1[i] rep
            i 2* x kt >>=  kt x += \   kt *= 1 + [1>>i]^2
            y 4 /4=  y kt -=  minY \   until kt>1
       until over c! 1+     kt x -= \ store rep, undo last
  loop                             \ -- zt1[gw/2]
  zt1 gw + 1+  swap                \ for i=gw/2 to gw
  do   1 i c!                      \   zt1[i]=1
  loop ;                           \ --

\ initialize and forget
 initzlrw initzl2                  \ linear tables init.
 initzrw                           \ trigo and hyper prep.
 zhrw zh2 1 initz2                 \ hyper tables init.
     init1/Kh  initln2  initln10/4 \ hyper constants init.
 ztrw zt2 0 initz2                 \ trigo tables init.
     initpi/4  init1/Kt  initrad   \ trigo constants init.
     initzt1             \ trigo reduction table init.

 forget init

EOF
\ ****************************************
\ To visualize the 3 CORDIC tables contents,
\ compile also this code and execute .TABLES

: .table        \ ad i --               \ dump table
  cr over body> >name .name             \ table name
  ." : key to dump/stop..." key drop    \
  gw 1+ swap                        \ for each table item
  do   key? if key key 2drop then       \ stop if key
       cr i 2 .r  dup @                 \ print item index
       dup h.  3 spaces               \ print mantissa in hex
       z swap :=  0 z reg@  e.      \ print item in E format
  2+ loop drop ;

: .tables
  zl2 1 .table                      \ dump linear table
  zh2 1 .table                      \ dump hyper  table
  zt2 0 .table                      \ dump trigo  table
;

EOF
```

****

Cordic constants initialization (part2)

This code is compiled, executed, and then forgotten; the side effect is to initialize the CORDIC tables and constants; for big values of SMB (more than 100), it may take a while, but this is why CORDIC is then faster than any other soft f-pack...

This initialization is done using only CORDIC routines: no "magic number" has to be meta-computed for CORDIC to work! That sounds a bit like FORTH meta-generation by FORTH itself, isn't it?

LISTING: FP_FORTH.FTH

```
\ ***********************************************
\ TurboForth CORDIC Floating Point package v1.0
\
\ CORDIC primitives high level definition
\ (see CORDICFP.FTH for files compilation order)
\ **********************************************

 decimal

: w@    \ addr -- w                 \ 16bit fetch, with
  dup >r c@ r> 1+ c@ 256 * + ;      \ low byte at low addr

: w!    \ w addr --                 \ 16bit store, with
  >r 0 256 um/mod r@ 1+ c! r> c! ;  \ low byte at low addr

: minZ  \ -- t/f                    \ true if *z >=0
  [ z rw + 2- ] literal w@          \ fetch z MS word
  0< not ;                          \ return opposite sign

: minY  \ -- t/f                    \ true if *x.*y <0
  [ x rw + 2- ] literal w@          \ fetch x MS word
  [ y rw + 2- ] literal w@          \ fetch y MS word
  xor 0< ;                         \ return composed signs

: clear \ r --                      \ *r=0
  rw erase ;                        \ clear register at r

: /4=   \ r n --                    \ *r=n/4  load immed.
  2048 *                            \ MSword : 0800 = 1/4
  swap rw 2-                        \ -- n*2048 r rw-2
  2dup erase                        \ clear except MSword
  + w! ;                            \ store n/4 at MSword

: +=    \ d s --                    \ *d+=*s  add s to d
  0 swap  rot  rw 2/ 0              \ -- cyIn s d  rw/2 0
  do  >r  tuck w@  0 tuck  d+       \ -- s+i cyIn+(s+i) cy
      r@ w@ 0  d+  swap r@ w!     \ -- s+i cyOut \ d+i=sum
      swap 2+  r> 2+                \ -- cy s+i+2 d+i+2
  loop
  2drop drop ;

: -=    \ d s --                    \ *d-=*s  sub s from d
  1 swap  rot  rw 2/ 0              \ -- cyIn s d  rw/2 0
  do  >r  tuck w@ not  0 tuck  d+   \ -- s+i cyIn+(s+i) cy
      r@ w@ 0  d+  swap r@ w!   \ -- s+i cyOut \ d+i=dif
      swap 2+  r> 2+                \ -- cy s+i+2 d+i+2
  loop
  2drop drop ;

: +/-   \ r --                      \ *r=-*r  negate r
  rw 0 skip  ?dup                   \ -- r+n rw-n \ 0 = -0
  if   dup 1 and                    \ n >= N even
       if  swap 1-  swap 1+  then   \ -- r+N rw-N
       >r dup w@ negate over w!     \ negate first word<>0
       r> 2/ 1                     \ (rw-n)/2-1 next words
       ?do  2+  dup w@ not over w!  \   are 1 complemented
```

```
      loop                               \ -- r+rw-2
  then drop ;

\ only null low words propagate carry, therefore only
\ the first non null word is negated, and the others
\ are 1 complemented

create 2^      \ increasing powers of 2
hex 0001 , 0002 , 0004 , 0008 , 0010 , 0020 , 0040 , 0080 ,
    0100 , 0200 , 0400 , 0800 , 1000 , 2000 , 4000 , 8000 ,
decimal

: >>=   \ n d s --                      \ *d=*s<<n   n=16Q+R
  swap  dup >r  - >r  negate 16 /mod >r \ -- R | Q s-d d
  2* 2^ + @                       \ -- r | Q s-d d \ r=2^R
  r> dup 0<
    \ n>0: shift right q=min(1-Q,rw/2):anti-underflow
  if  not  rw 2/ min  dup 2* r> + >r     \ -- r q | s-d+2q d
      rw 2/ over -  r>  swap  r>  swap  \ -- r q s-d+2q d N
      dup >r 0                          \          N=rw/2-q
      ?do  2dup + @  over !  2+          \ s+2q d rw-2q cmove
      loop                            \ -- r q S D | N \D=d+2N
      tuck  dup >r  + 2- @ 0< >r        \ -- r q D | sign D N
      swap 2* r@ fill                 \ d+rw-2q 2q sign fill
      dup r> um* drop  r>  r> 0         \ -- r h D  N 0
      ?do  2- >r  over  r@ @  um*        \ -- r h L H
           >r swap r> or  r@ !  r>     \ -- r L D \ H+h at D
      loop                              \ -- r L d
  else                    \   n<0:   shift   left
q=min(Q,rw/2):anti-overflow
      rw 2/ min  r> over 2* - >r        \ -- r q | s-d-2q d
      rw 2/ over - r> swap r> rw + swap
       \ -- r q s-d-2q d+rw | N
      dup >r 0                          \      N=rw/2-q
      ?do  2-  2dup + @  over !         \ s d+2q rw-2q cmove>
      loop nip                        \ -- r q D | N \ D=d+2q
      swap 2*  2dup -  swap  erase \ -- r D | N \ d 2q eras
      0 swap  r> 0                      \ -- r h D  N 0
      ?do  >r  over  r@ @  um*          \ -- r h L H | D
         >r or r> swap  r@ ! r> 2+ \ -- r H D+2 \ L+h at D
      loop                              \ -- r H d+rw
  then
  drop 2drop ;  \ --

: :=    \ d s --                     \ copy sce s into dest d
  rw 2/  0                              \ -- d s  rw/2 0
  do     dup >r w@  over w!  2+  r> 2+   \ (d+2i) = (s+2i)
  loop  2drop ;

: >>    \ d --                       \ divide register d by 2
  rw +  dup 2- w@ 0<  rw 2/ 0           \ -- d+rw H  rw/2 0
  do  >r  2- dup w@  tuck 2/  r> 1 and  \ -- L D L/2 cy
      if   32768 or                    \ if cy set bit15 of L/2
      else 32767 and                    \ else  reset bit15
      then                              \ -- L D L>>1
      over w! swap                      \ -- D L  \ L=newH
  loop 2drop ;

\ normalize float with mantissa at addr r and binary
\ exp. e; return corrected exponent e'.
: norm  \ e r -- e'
  dup >r  rw +  dup 2- w@ 0< >r       \ -- e r+rw    | sign r
  begin  2-  dup w@  r@ <>  until     \ -- e r+rw-n | sign r
  dup w@  r> if not then              \ -- e r+rw-n MSW  | r
  swap r@ -  8 *                      \ -- e MSW rw-n    | r
  [ rw 8 * 3 - ] literal -  over 0<  \ -- e MSW 3-8n 0< | r
  if    16 +                          \ if MSW<0: MSbit=bit15
  else  begin 1+ swap 2/ tuck 0= until
        \ else bs=3-8n+MSb_rank
  then  nip                              \ -- e bs | r
  dup abs SMB >                          \ if |bs|>SMB:
  if    2drop  -16384  r> clear       \ zero: 000000<<-16384
  else  dup r> dup >>=  +              \ -- e+bs \ *r <<= -bs
  then ;                               \ -- e'

EOF
******************************************
```

This implementation is given for MACHINE INDEPENDANCE and as an algorithmic template for assembler versions (see FP_8086.FTH version in 8086 series assembler)

W@ and W! are Intel-like versions of @ and !; this memory model (byte significance increasing as addresses) is the most convenient and efficient for primitives (this is also the one chosen by IEEE standards)

MINY and MINZ return registers signs; cordic rotation direction is determined at each step by minZ for ROTATING, or by minY for VECTORING

Registers are either cleared (with CLEAR)
- or loaded with immediate values (with /4=)
- or loaded from other registers (with >>= or :=)
- or loaded from the stack (with REG! --see FPMAIN.FTH)
- or added or subtracted with other registers (with += or -=)
- or negated (with +/-)

Cordic efficiency relays mainly on the >>= shifting routine :
source mantissa at addr s is transfered at destination addr d
while shifted n times

```
  >>=         \ n d s --  \ right for n>0
  >>=         \ n d s --  \ left  for n<0
optimized routines are also given
  :=          \ d s --     \ copy s into d ( >>= for n=0 )
  >>          \ d --       \ divide d by 2 ( >>= for n=1 )
```

Normalized mantissae are in the range [-1..-1/2[U [1/2..1[(-1 and 1/2 included, -1/2 and 1 excluded)
- positive normalized mantissae MShexDigit = 1
- negative normalized mantissae MShexDigit = E

Is considered null any mantissa with MSbit<lsb (ie which must be normalized by more than SMB left shifts)
Zero is coded with mantissa=0 and exp=-16384

LISTING: FP_8086.FTH

```
\ ***********************************************
\ TurboForth CORDIC Floating Point package v1.0
\
\ CORDIC primitives high level definition
\ (see CORDICFP.FTH for files compilation order)
\ **********************************************

ALSO

CODE w@ \ adr -- w                    \ 16bit fetch with
  BX pop  0 [BX] push                 \ low byte at low addr
  next  end-code

CODE w! \ w adr --                    \ 16bit store with
  BX pop  0 [BX] pop                  \ low byte at low addr
  next  end-code

CODE minZ       \ -- t/f              \ true if *z>=0
  AX AX xor                           \ AX=0
  AL z rw 1- + #) cmp                 \ if *zMSB>=0 :
  0>= if  AX dec  then                \     AX=-1
  1push  end-code                     \ -- AX

CODE minY       \ -- t/f              \ true if *y.*x<0
  AX AX xor                           \ AX=0
  x rw 1- + #) DL mov                 \ DL=*xMSB
  y rw 1- + #) DL xor                 \ if (*yMSB:+:*xMSB)<0
  0< if  AX dec  then                 \     AX=-1
  1push  end-code                     \ -- AX

CODE clear      \ r --                \ *r=0  clear r
  AX AX xor  DI pop                   \ AX=0  DI=r (ES=DS)
  rw 2/ # CX mov  rep AX stos         \ erase r to r+rw-2
  next  end-code                      \

CODE /4=        \ r n --              \ *r=n/4  load immed.
  AX pop  DI pop                      \ AX=n  DI=r (ES=DS)
  11 # CL mov  AX CL shl              \ MSword: 0800 = 1/4
  rw 2- # CX mov  rep byte AL stos  AX stos
                                      \ clear r, r+rw-2 := AX
  next  end-code

CODE +=         \ d s --              \ *d+=*s
  SI DX mov                           \ save SI (ES=DS)
  SI pop  BX pop  SI BX sub  2 # BX sub \ SI=s  BX=d-s-2
  clc  rw 2/ do  AX lods  AX 0 [BX+SI] adc
                                   \ for i=0 to rw/2 step 2
  loop                               \   (d+i) += AX = (s+i)
  DX SI mov  next  end-code           \ restore SI

CODE -=         \ d s --              \ *d-=*s
  SI DX mov                           \ save SI (ES=DS)
```

```
 SI pop  BX pop  SI BX sub  2 # BX sub \ SI=s  BX=d-s-2
 clc  rw 2/ do  AX lods  AX 0 [BX+SI] sbb
                                   \ for i=0 to rw/2 step 2
 loop                              \   (d+i) -= AX = (s+i)
 DX SI mov  next  end-code         \ restore SI

CODE +/-        \ r --             \ *r=-*r
 BX pop    2 # BX sub              \ BX=r-2 (to save ZF)
 rw 2/ do  2 # BX add  0 [BX] neg  \ negate words
 loope                             \ until no carry or end
 HEX E3 DECIMAL ( cx<>0 )         \ if not end of register
 if  here  2 # BX add  0 [BX] not  \   complement words
     loop                          \   until end of reg.
 then
 next end-code

CODE >>=        \ n d s --         \ *d=*s>>n
 BX pop  DI pop  DI BX sub  DX pop \ BX=s-d  DI=d  DX=n
 rw 1- # AX mov                    \ AX=rw-1
 HEX  0703 # CX mov  DECIMAL       \ CH=7  CL=3
 DH DH or  0<                      \ if n<0 : left shifts
 if  DX neg  std                   \ |n|=8q+r
     DL CH and  DX CL sar  CH CL mov   \ DX=q  CL=r
     AL CH mov  DL CH sub  <=      \ CH=rw-1-q
     if  AX DX mov                \ if q+1 >= rw : DX=rw-1
         0 [BX+DI] AX mov          \   AX := (s)
         AX CL shl                 \   AX<<r
         DX DI add                 \   DI=d+rw-1
     else                          \ else (rw>q+1) :
         DX BX sub  BX dec         \   BX=s-d-q-1
         AX DI add                 \   DI=d+rw-1
         begin  0 [BX+DI] AX mov   \   for i=rw-1-q dnto 1
                AX CL shl          \     AX := (s+i-1)<<r
                AH AL xchg  byte AL stos
                \     (DI=d+q+i) := AH
                CH dec  0=         \
         until  AH AL mov          \   AL=MSByte  DI=d+q
     then  AH AH xor               \ AH=0
 else                              \ n>=0: right shifts
     gw # DX cmp  >=               \ if n>=gw (underflow):
     if    AX DX mov  AX AX xor    \    DX=rw-1 AX=0 else:
     else  DL CH and  DX CL sar    \ CH=r   DX=q    n=8q+r
           DX BX add  CH CL mov    \ BX=s-d+q       CL=r
           AL CH mov  DL CH sub    \ CH=rw-1-q
           0<>                     \ for i=0 upto rw-q-2
           if  begin  0 [BX+DI] AX mov \  AX := (s+q+i)
                      AX CL sar    \  AX>>r
                      byte AL stos \  (d+i) := AL
                      CH dec  0=   \
               until               \ DI=d+rw-q
           then                    \
           0 [BX+DI] AL mov  cbw   \ AL=(s+rw-1)
           AX CL sar               \ AX>>r  DX=q
     then                          \
 then                              \ (see AX,DI,DX)
 byte AL stos                      \ store last signif.byte
 AH AL mov  DX CX mov  rep byte AL stos
                                   \ fill remaining bytes
 cld  next  end-code

CODE :=         \ d s --           \ copy register s into d
 SI DX mov                         \ save SI (ES=DS)
 SI pop  DI pop                    \ SI=s  DI=d
 rw 2/ # CX mov  rep movs          \ move rw/2 words
 DX SI mov  next  end-code         \ restore SI

CODE >>         \ r --             \ shift right once
 BX pop   rw 2- # BX add  0 [BX] sar   \ shift MSword
 rw 2- 2/ do
     BX dec  BX dec  0 [BX] rcr    \ other words
 loop
 next  end-code

CODE norm       \ e r -- e'        \ see forth version
 BX pop  DX pop                    \ BX=r      DX=e
 rw # CX mov  BX DI mov  CX DI add \ CX=rw     DI=r+rw
 DI dec  0 [DI] AL mov  cbw        \ AH=sign(r)
 AH AL mov  std  repz byte scas  cld
  \ look back for MSByte
 1 [DI] AL mov                     \ CX=rw-n   AL=MSByte
 AH AH or  0< if  AX not  then     \ AX = |MSByte|
 CX shl  CX shl  CX shl            \ CX=8rw-8n
 rw 8 * 3 - # CX sub               \ CX=3-8n
 begin  CX inc  AX shr  0=         \ (with SHR, loop ends)
 until                             \ CX=de=3-8n+MSbit_rank
 SMB negate # CX cmp  <=           \ zero if |de|>SMB
 if     -16384 # DX mov            \ zero: 000000<<-16384
        DX push  BX push           \ -- zero_exp r
        ' clear >body #) jmp       \ call CLEAR to clear r
 then                              \
 CX DX add  DX push                \ -- e'      e'=e+de
 CX push  BX push  BX push         \ -- e' de r r
 ' >>= >body #) jmp                \ call >>= to shift r
 end-code                          \ de>gw: >>= is CLEAR

PREVIOUS
EOF
```

****

This implementation is dedicated to 8086 series processors (see FP_FORTH.FTH version as high level template for other assemblers)

W@ and W! are Intel-like versions of @ and !
 this memory model (byte significance increasing as addresses)
 is the most convenient and efficient for primitives (this is also the one chosen by IEEE standards)

MINY and MINZ return registers signs; cordic rotation direction is determined at each step by minZ for ROTATING, or by minY for VECTORING

Registers are either cleared (with CLEAR)
 or loaded with immediate values (with /4=)
 or loaded from other registers (with >>= or :=)
 or loaded from the stack (with REG! --see FPMAIN.FTH)
 or added or subtracted with other registers (with += or -=)
 or negated (with +/-)

Cordic efficiency relays mainly on the >>= shifting routine:
 source mantissa at addr s is transfered at destination addr d
 while shifted n times

```
   >>=        \ n d s --     \ right for n>0
   >>=        \ n d s --     \ left  for n<0
 optimized routines are also given
   :=         \ d s --     \ copy s into d ( >>= for n=0 )
   >>         \ d --       \ divide d by 2 ( >>= for n=1 )
```

NORM is executed for each result returned on the fpstack
 (see REG@ in FPMAIN.FTH)
 Normalized mantissae are in the range [-1..-1/2[U [1/2..1[
  (-1 and 1/2 included, -1/2 and 1 excluded)
  positive normalized mantissae MShexDigit = 1
  negative normalized mantissae MShexDigit = E
 Is considered null any mantissa with MSbit<lsb
  (ie which must be normalized by more than SMB left shifts)
 Zero is coded with mantissa=0 and exp=-16384

****

```
\ version for processor with fast multiply um*
\ this is a multiprecision fixed point multiply

variable carry           \ for multiple precision arith.

: clc     \ --           \ clear carry flag
  carry off  ;

CODE +c   \ n1 n2 -- n1+n2+cy
  AX pop  BX pop  byte carry #) rcr   \ prepare data
  BX AX adc   byte carry #) rcl       \ add, store carry
  1push  end-code                     \ return result


: y=xz  \ e -- e'                     \ y<<e' = x*z<<e
  [ z rw + 2- ] literal @ 0< dup      \ use z modulus
  if  z +/-  then                     \
  [ x rw + 2- ] literal @ 0< dup      \ use x modulus
  if  x +/-  then                     \
  xor                               \ -- s \ s = sign(y)
  -3 z dup >>=
  0 0 0  z rw +  x
  rw 2/ 0                             \
  do rw 2/ i
     do dup 2+ >r @  swap 2- dup >r @ \ @++ >r swap --@ >r
```

```
      um*  -rot clc +c >r +c >r 0 +c  \ -- s cy hi+H lo+L
      r> r> r> r>                     \ -- s cy hi lo a- a+
   loop                               \
   rot  i 1- 2* 0 max y + !           \ store lo at y+2(i-1)
   >r >r  0 -rot  r> r> swap 2+       \
 loop  2drop                          \ (newhi=cy newlo=hi)
 [ y rw + 2- ] literal !  2drop       \ -- s \ store last hi
 if  y +/-  then  ;                   \ --
```

LISTING: FPKERNEL.FTH

```
\ ***********************************************
\ TurboForth CORDIC Floating Point package v1.0
\
\ CORDIC main functions and user interface
\ (see CORDICFP.FTH for files compilation order)
\ ***********************************************

\ ********************************************
\                    STACK
\ ********************************************
\ fp stack grows towards low memory, from fp0@, max.depth=9
\ each cell holds a 16bits exponent in 2's complement
\ format, followed by a mantissa in register format
\ (rw bytes)

rw 2+ constant sw            \ stack width = exp + mantissa

variable fp  9 sw * allot    \ fp stack top pointer + stack

here constant fp0@             \ bottom of fp stack

: fp0!  \ --                   \ clear fp stack
  fp0@ fp ! ;

fp0@ fp !                      \ initialize fp stack

: ?ferror      \ n --
  swap if  fp0!  case
    0 of " empty fp stack"                 endof
    1 of " full fp stack"                  endof
    2 of " conversion overflow"            endof
    3 of " overflow"                       endof
    4 of " underflow"                      endof
    5 of " divide: arg=0"                  endof
    6 of " logarithm: arg=0"               endof
    7 of " logarithm: arg<0"               endof
    8 of " square root: arg<0"             endof
    9 of " asin/acos/atanh: |arg|>1."      endof
   10 of " use F# inside colon definitions"  endof
       " unknown fp error" rot
  endcase  true ?error
  else drop then ;

: (fp@+)        \ n -- addr        \ addr is n bytes
  fp @ +                            \ below top of stack
  [ fp0@ sw - ] literal
  over u< 0 ?ferror ;               \ check stack underflow

: (fpop)        \ -- addr           \ addr is top of stack
  0 (fp@+)  sw fp +! ;              \ before a pop

: (fpush)       \ -- addr           \ addr is top of stack
  fp @ sw -                         \ after a push
  dup [ fp 2+ ] literal
  u< 1 ?ferror                      \ check stack overflow
  dup fp ! ;

: (fpick) \ n --                    \ pick sw bytes,
  (fp@+) (fpush) sw cmove ;         \ n bytes under top

: (froll) \ n --                    \ roll sw bytes,
  dup (fpick)                       \ n bytes under top
  fp @  dup sw +  rot sw +  cmove>
  sw fp +! ;

: fpick \ n -- | fn..f0 == fn..f0 fn    \ pick nth float
  sw * (fpick) ;                         \ (top is 0th)

: froll \ n -- | fn f..f0 == f..f0 fn   \ roll nth float
  sw * (froll) ;                         \ (top is 0th)

: fdrop \ f ==
  (fpop) drop ;
```

```
: fdup  \ f == f f
  0     (fpick) ;

: fover \ f2 f1 == f2 f1 f2
  sw    (fpick) ;

: fswap \ f2 f1 == f1 f2
  sw    (froll) ;

: frot  \ f3 f2 f1 == f2 f1 f3
  sw 2* (froll) ;

: reg!  \ f == | r -- e      \ f=*r<<e   stack -> register
  (fpop)                     \ -- r top  \ fp-=sw
  tuck 2+ swap rw cmove      \ -- top    \ *r=(top+2)
  @ ;                        \ -- e

: reg@  \ e r -- | == f      \ f=*r<<e   register -> stack
  tuck norm                  \ -- r e'
  dup -16384 < 4 ?ferror     \ check exp. underflow
  dup  16383 > 3 ?ferror     \ check exp. overflow
  (fpush)                    \ -- r e' top  \ fp+=sw
  tuck !                     \ -- r top     \ (top+0)=e'
  2+ rw cmove ;              \ --           \ (top+2)=*r

: (nearest)     \ r n --               \ round r to bit n
  xs 2 /4=  xs dup >>=  xs += ;        \ r+=(1/2)>>n

defer round                            \ round is used by
  ' (nearest) is round                 \ SINT DINT F!

: nearest                              \ set rounding
  ['] (nearest) is round ;             \ to nearest

: truncate                             \ set rounding
  ['] drop is round ;                  \ to -infinity

: sint  \ f == | -- n                  \ float-to-single
  x reg!  dup 15 > 2 ?ferror           \ -- e  check overflow
  13 swap - x dup >>=  x 13 round     \ align comma and round
  [ x rw 2 - + ] literal @  ;          \ fetch high single

: dint  \ f == | -- d                  \ float-to-double
  x reg!  dup 31 > 2 ?ferror           \ -- e  check overflow
  29 swap - x dup >>=  x 29 round     \ align comma and round
  [ x rw 4 - + ] literal               \ fetch high double
  dup @  swap 2+ @  ;                 \ 2@ is processor spec.

: s>f   \ n -- | == f                  \ single-to-float
  zs clear
  [ zs rw 2 - + ] literal  !           \ store high single
  13 zs reg@ ;                         \ normalize and push

: d>f   \ d -- | == f                  \ double-to-float
  2dup dabs swap 0< or                 \ too big for s>f ?
  if    zs clear
        [ zs rw 4 - + ] literal        \ store high double
        tuck 2+ !  !                   \ 2! is processor spec.
        29 zs reg@                     \ normalize and push
  else  drop s>f
  then ;

: fdup0=        \ f == f | -- f=0      \ non destructive f0=
  fp @ @ -16383 < ;                    \ zero: f <= 2^-16384

: fsign         \ f == f | -- f<0      \ non destructive f0<
  fp @ rw + @  0< ;                    \ MSword at fp+2+rw-2

: f0=           \ f == | -- f=0
  fdup0= fdrop ;

: f0<           \ f == | -- f<0
  fsign fdrop ;

: f0>           \ f == | -- f>0
  fdup0= f0< or not ;

: xy! \ f2 f1 == | -- e                \ f2=*x<<e  f1=*y<<e
  y reg!  x reg!                       \ -- ey ex
  2dup <                               \ -- ey ex ey<ex
  if    dup rot - y                    \ -- ex ex-ey y
  else  over swap - x                  \ -- ey ey-ex x
  then                                 \ -- e   de   r
  dup >>= ;                            \ -- e     *r>>=de
```

```
: f+    \ f2 f1 == f2+f1            \ fp addition
  xy!  x y +=  x reg@ ;

: f-    \ f2 f1 == f2-f1            \ fp subtraction
  xy!  x y -=  x reg@ ;

: fnegate       \ f == -f
  zs reg!  zs +/-  zs reg@ ;

: fabs          \ f == abs(f)
  fsign                             \ == f   | -- f<0
  if   fnegate                      \ == -f
  then ;                            \ == abs(f)

: fmax  \ f2 f1 == max(f2,f1)
  fover f- fsign                    \ == f2 f2-f1 | -- f2<f1
  if    fdrop                       \ == f2
  else  f+                          \ == f1
  then ;                            \ == max(f2,f1)

: fmin  \ f2 f1 == min(f2,f1)
  fover f-  fsign                       \ == f2 f2-f1 | --
f2<f1
  if    f+                          \ == f1
  else  fdrop                       \ == f2
  then ;                            \ == min(f2,f1)

: f=    \ f2 f1 == | -- f2=f1
  f- f0= ;

: f<    \ f2 f1 == | -- f2<f1
  f- f0< ;

: f>    \ f2 f1 == | -- f2>f1
  f- f0> ;

\ *****************************************
\                  MEMORY
\ *****************************************
\ floats are stored in memory in the same format as stack :
\ msb-bit16 = normalized mantissa in 2's complement
\             comma left of bit msb-3
\             (mshexdigit:  1:positive  E:negative  0:zero)
\ bit15-lsb = binary exponent in 2's complement

\ see also IEEE memory ops F32! F32@ F64! F64@

: f!    \ addr -- | f ==
  (fpop) swap sw cmove ;

: f@    \ addr -- | == f
  (fpush) sw cmove ;

: f,          \ f ==                \ fp version of ,
  here f!  sw allot ;

: flit        \ == f :exec;         \ DON'T interpret FLIT
  r>  dup f@  sw + >r ;

: fliteral      \ f == :comp;       \ compile fp literal
  compile flit  f, ;                \ NOT state smart
immediate

: fconstant    \ comp: f ==         \ fp constant
               \ exec: == f
  create f,
  does>  f@ ;

: fvariable    \ comp: --           \ fp variable
               \ exec: -- ad
  create sw allot ;

: f*array       \ comp: wn..w1 n -- \ multi-dimensionnal
                \ exec: in..i1 -- ad
                \ array of fp variables
  create
    dup 1 < abort" No. of dim.< 1"
    depth over <= 1 ?error          \ stack underflow
    dup c,   sw  swap 0             \ -- wn..w1 sw  n 0
    do   over 1 < abort" one dim.< 1" \ -- wn..wi N
         over ,                     \ compile ith dimension
         *                          \ -- wn.. wi*N
    loop                            \ -- N
    allot                           \ allot wn*..*w1*sw
```

```
  does>                                \ -- in..i1 addr
    0 swap  count 0                    \ -- in..i1 0 addr+1 n
    do                                 \ -- in..ii N ad
         dup 2+ >r @                  \ -- in..ii N dim | ad+2
         dup 3 pick <=                 \ comment this and next
         abort" index out of range"
         \ to improve efficiency
         * +  r>                       \ -- in.. ii+N*dim ad+2
    loop                               \ -- N addr0th
    swap sw * +  ;                     \ -- addrNth

\ *****************************************
\         CONVERGENCE DIRECTION
\ *****************************************

defer minYZ     \ -- t/f           \ either minY or minZ

: rotating      \ --               \ set rotating
  y clear  ['] minZ is minYZ ;

: vectoring     \ --               \ set vectoring
  z clear  ['] minY is minYZ ;

: m.     \ ad --                   \ mantissa hex dump
  base @ >r hex  3 spaces  dup rw + 2-
  \ MS byte at high add
  do   i @  0 <# # # # # #> type
  -2 +loop
  r> base ! ;                      \ (used by f.s)

\ *****************************************
\                    TRACE
\ *****************************************
\ comment this section not to compile the trace
\ and look for TRACE instructions to comment

: r.    \ r --                     \ register hex dump
  dup m. ." =" body> >name .name ; \ along with its name

: xyz.  \ i --                     \ dump at step i
  cr  3 .r  x r. y r. z r.  minYZ 3 .r  \ x y z and minYZ
  ." =" ['] minYZ >body @ >name .name
  key 3 = abort"  ^C " ;           \ abort if control-C

defer trace
' drop is trace

: tron                             \ set trace on
  ['] xyz. is trace ;

: troff                            \ set trace off
  ['] drop is trace ;

: tr'                              \ trace only following
  tron ' execute troff ;           \ word in input buffer

\ *****************************************
\                    LINEAR
\ *****************************************

: linear        \ --               \ Linear complete Rot.
  zl2  gw 1+ 1                     \ for i=1 to gw
  do   z over @  y x  minYZ
  \   i trace \ select rot.direction
       if    += -=                 \   y+=x  z-=zlrw[i]
       else  -= +=                 \   y-=x  z+=zlrw[i]
       then                        \ -- zl2[i]
         x     >>   2+             \ -- zl2[i+1]  \ x>>1
  loop drop                  \ gw trace \ --
;                                  \ with: zlrw[i] = 1>>i

\ x --> 0                              x --> 0
\ y --> y+(2x*z)                                   y --> 0
{vectoring}
\ z --> 0 {rotating}                   z --> z+(y/2x)

\ this routine may be already defined in the
\ primitives file for processor with fast multiplier

: y=xz  \ e -- e'                 \ y<<e' = xz<<e
  rotating linear 1- ;

: f*    \ f2 f1 == f2*f1           \ float multiplication
  z reg!  x reg!  +               \ -- e2+e1 \ x=m2 z=m1
```

```
  y=xz  y reg@ ;                      \ --       \ y=2*m2*m1

: f/    \ f2 f1 == f2/f1              \ float division
  fdup0= 5 ?ferror                    \ check divide by zero
  x reg!  y reg!  - negate            \ -- e2-e1 \ x=m1 y=m2
  vectoring linear  1+ z reg@ ;       \ --        \ z=m2/(2*m1)

: f1/   \ f == 1/f                    \ float inverse
  fdup0= 5 ?ferror                    \ check divide by zero
  x reg! negate  y 2 /4=              \ -- -e     \ x=m  y=1/2
  vectoring linear  2+ z reg@ ;       \ --        \ z=1/(4*m)

: (r/mod)      \ f == | eX mX -- Q ey  \ f = m<<e = Q*X+R
               \ vectoring \ (for trace) \ X=mX<<eX  R=y<<ey
  x swap :=  1-  y reg!  z clear      \ -- eX-1 e \ x=mX y=m
  dup >r -  zl2  over 0<              \ -- eX-1-e zl2 eX<=e| e
  if  over negate 0                   \ if e>=eX :
      do  z over @  y x minY
          \ i trace \   do the e-eX first
          if    +=  -=               \   division steps,
          else  -=  +=               \   leaving :
          then                       \   z = integer quotient
            x      >>    2+          \   y = fract. remainder
      loop                           \ -- eX-1-e zl2[e-eX]| e
  then                \ over trace \ -- eX-1-e ad         | e
  \ now  adjust  yz  to  force  remainder*dividend>0  (x=div
y=rem)
  0 y norm  dup -16384 >             \ -- eX-1-e ad ey y<>0|e
  if  negate y dup >>=  minY         \ if y<>0 and yx<0 :
      if  over 0<                    \   if e>=eX, restore
          if   x dup +=  2-          \     last division step
          else swap 1+               \   else  -- zl eX-e | e
               dup y dup >>=         \     re-align y to x
               r> + >r  -1 swap      \     -- -1 zl | eX
          then                       \   -- ez ad | ey
          y x +=  z over @ -=        \   repeat last div.step
      then  dup                      \   balance stack
  then    2drop                      \ -- ez | ey
  13 + z dup >>=                     \ align z comma on word
  [ z rw + 2- ] literal @
  \ -1 trace \ fetch integer quotient
  r> ;                               \ -- Q ey

: (z/mod)      \ f == | eX mX -- Q  \ f = Q*X+R
  (r/mod)  negate z y >>= ;          \ X=mX<<eX  -X<z=R<X

: f/mod      \ f2 f1 == R | -- Q     \ f2 = Q*f1+R (Q int)
  fdup0= 5 ?ferror                   \ check divide by zero
  x reg!  x (r/mod)  y reg@ ;

: ffrac          \ f2 f1 == R        \ fp remainder of f2/f1
  f/mod drop ;

: fint           \ f == fi           \ fp integer part of f
  fsign dup                          \ f = s*m<<e
  if   fnegate                       \ use modulus of f
  then                               \ -- s | == m<<e
  fp @ @  dup gw <                   \ -- s e e<gw | == m<<e
  if  dup 0>                         \ if 0<e<gw
      if   dup [ rw 8 * 3 - ] literal -
      \   clear m fract. bits
           fp @ 2+  over negate over \ by shifting 8rw-3-e
           dup >>=  dup >>=          \   times right and left
      else -16384 fp @  dup 2+ clear  ! \ else store zero
      then                           \ -- s e | == abs(fi)
  then  drop                         \ -- s | == abs(fi)
  if   fnegate                       \ restore sign
  then ;                             \ == fi

\ ****************************************
\           HYPERBOLIC
\ ****************************************

: (hyper)       \ ad i -- ad         \ Hyper elem. Rotation
  dup  xs y >>=  ys x >>=            \ xs=y>>i  ys=x>>i
  z over @  y ys  x xs  minYZ        \ select rot.direction
  if    += += -=                     \   x+=xs y+=ys z-=(ad)
  else  -= -= +=                     \   x-=xs y-=ys z+=(ad)
  then ;                             \ with: (ad)=Atanh(1>>i)

: hyper          \ --                \ Hyper complete Rot.
  4 zh2  gw 1+ 1                     \ for i=1 to gw
  do   over i =                      \   -- k zh2[i] i=k
       if  >r  3 * 1+  r>            \      i=4,13..k,3k+1..
           i (hyper)
           \  i trace \     repeat step
       then                               \   -- k zh2[i]
       i (hyper)  2+      \  i trace \   -- k zh2[i+1]
  loop 2drop              \ gw trace \ --
;

\ x --> Kh*[x*ch(z) + y*sh(z)]       x --> Kh*sqr(x*x-y*y)
\ y --> Kh*[y*ch(z) + x*sh(z)]       y --> 0    {vectoring}
\ z --> 0           {rotating}       z --> z+Atanh(y/x)

: (lnsr)       \ --                  \ entry: z=m
  x 4 /4=  x z +=  y -4 /4=  y z +=  \ x=m+1 y=m-1
  vectoring hyper ;                  \ x=2Kh.sqr(m) z=ln(m)/2

: fsqr   \ f == sqr(f)               \ float square root
  fsign 8 ?ferror                    \ check sqr of f<0
  fdup0= not                         \ if f=0: sqr(f)=0.
  if   z reg!                        \ -- e  \ z=m  f=m<<e
       dup 1 and                     \ if e odd :
       if   1-  z dup +=             \   e-=1  z*=2
       then 2/                       \ -- e/2
       (lnsr)  2 z 1/Kh >>=          \ x=2Kh*sqr(m) z=1/4Kh
       y=xz 1+  y reg@               \ y=sqr(m)
  then ;                             \ sqr(f)=sqr(m)<<(e/2)
\ sqr(m<<e)=sqr(m)<<(e/2)
\    sqr(m) = sqr[(m+1)^2 - (m-1)^2]/2

: 2z+e*ln2      \ e -- | == f        \ z=m --> f=2*m+e*ln(2)
  dup 0=                             \ -- e e=0
  if   1+ z reg@                     \ if e=0: == 2m
  else x clear [ x rw + 2- ] literal !  \ x=e>>13
       13 x norm  zs z :=  z ln2 :=
       \ -- n \ x=e>>n zs=m z=ln2
       y=xz  dup 1- zs dup >>=       \ -- n \ y=x*z zs=2m>>n
       y zs +=  y reg@               \ y = 2m>>n + ln2*e>>n
  then ;                             \ == 2m+e*ln2

: fln   \ f == ln(f)                 \ fp natural logarithm
  fsign 7 ?ferror  fdup0= 6 ?ferror \ check log of f<=0
  z reg!  (lnsr)  2z+e*ln2 ;
  \ ln(m<<e)=ln(m)+e*ln(2)

\ ln(m<<e)=ln(m)+e*ln2   ln(m)  = 2*Atanh[(m-1) / (m+1)]

: (exp)        \ f == | -- Q         \ f=Q*ln(2)+R
  0 ln2 (z/mod)  x 1/Kh :=           \ -- Q \ x=1/Kh  z=R
  rotating hyper  x y += ;           \ -- Q \x=exp(R) y=sh(R)

: faln  \ f == exp(f)                \ fp exponential
  (exp) x reg@ ;                     \ f=Q*ln(2)+R
                                     \ exp(f) = exp(R)<<Q

\ exp(Q*ln2+R)=exp(m)<<Q     exp(m) = sh(m) + ch(m)
\ sh(Q*ln2+R)=[exp(R)-exp(-R)>>2Q]<<(Q-1)
\ ch(Q*ln2+R)=[exp(R)+exp(-R)>>2Q]<<(Q-1)

: ln10  \ == 2.3025851...            \ natural log of 10
  2 ln10/4 reg@ ;

: flog  \ f == log(f)                \ decimal logarithm
  fln ln10 f/ ;

: falog \ f == 10^f                  \ decimal power
  ln10 f* faln ;

: f**   \ f2 f1 == f2^f1             \ fp power
  fswap fln f* faln ;

: (scth)       \ f == | -- Q-1       \ f=Q*ln(2)+R
  (exp)   y dup +=  y x -=           \ -- Q    y=-exp(-R)
  dup 2* y dup >>=                   \ -- Q    y=-exp(-R)>>2Q
  x y -=  y dup +=  y x +=  1- ;     \ -- Q-1 x=ch(f)>>(Q-1)
                                     \          y=sh(f)>>(Q-1)
: fsinh        \ f == sh(f)          \ hyperbolic sine
  (scth)  y reg@ ;

: fcosh        \ f == ch(f)          \ hyperbolic cosine
  (scth)  x reg@ ;

: ftanh        \ f == th(f)          \ hyperbolic tangent
  (scth) drop  vectoring linear      \ z=sh(f)/2ch(f)
  1 z reg@ ;

: (?1-) \ f == | -- f<0 E            \ exit: |f|=-zs=1-z>>E
  fsign dup not                      \ -- s f>=0 | == f
```

```
  if  fnegate                          \ use negative f modulus
  then                                 \ -- s | == -|f|
  z reg!  z SMB (nearest)  z norm      \ -- s e \ round 1+eps
  dup 0> 9 ?ferror                     \ -- s e \ check |f|>1
  negate zs z >>=  z 4 /4= z zs +=     \ -- s \ zs=-|f| z=1-|f|
  0 z norm negate ;                    \ -- s E \ z=(1-|f|)<<E

: (ath) \ E -- E                       \ entry:0<z<1 |f|=1-z>>E
  y 8 /4=  y z -=  x 8 /4=  x z +=     \ y = 2-z  x = 2+z
  dup z dup >>=  y z -=  x z -=        \ y-=z>>E  x-=z>>E
  vectoring hyper ;                  \ x=Kh*sqr(1-f^2)<<(E/2+1)
                                     \ z=Atanh|f|-(E/2)*ln(2)

: fatanh        \ f == Atanh(f)        \ inverse hyp. tangent
  fdup0= not                           \ Atanh(0)=0.
  if (?1-) (ath) 2z+e*ln2              \ -- f<0 | == 2*Atanh|f|
     if   fnegate                      \ Atanh(-z) = -Atanh(z)
     then                              \ == 2*Atanh(f)
     -1 fp @ +!                        \ == Atanh(f)
  then ;

\ Atanh(1-m>>E) = Atanh[(2-m-m>>E) /
\     (2+m-m>>E)]+(E/2)*ln(2)

\ ****************************************
\            TRIGONOMETRIC
\ ****************************************

: pi          \ == 3.141592654...      \ how many digits do you
  2 ztrw reg@ ;                        \ want ? Try SMB=100

defer rad>   \ radians == units        \ used by inverse trigo
  ' noop is rad>                       \ functions

defer >rad   \ units == radians        \ used by direct trigo
  ' noop is >rad                       \ functions

: radians    \ --                      \ set radians as default
  ['] noop is rad>                     \ unit for trigo funct.
  ['] noop is >rad ;

: rad>deg    \ radians == degrees      \ multiply by 59.296...
  6 r>d reg@ f* ;

: deg>rad    \ degrees == radians      \ multiply by 0.01745...
  -5 d>r reg@ f* ;

: degrees    \ --                      \ set degrees as default
  ['] rad>deg is rad>                  \ unit for trigo funct.
  ['] deg>rad is >rad ;

: rad>grd    \ radians == grades       \ multiply by 63.662...
  6 r>g reg@ f* ;

: grd>rad    \ grades == radians       \ multiply by 0.01571...
  -5 g>r reg@ f* ;

: grades     \ --                      \ set grades as default
  ['] rad>grd is rad>                  \ unit for trigo funct.
  ['] grd>rad is >rad ;

: (trigo)    \ ad i -- ad              \ Trigo elem. Rotation
  dup  xs y >>=  ys x >>=              \ *xs=y>>i  *ys=x>>i
  z over @  y ys  x xs  minYZ          \ select rot.direction
  if    -= += -=                       \   x-=xs y+=ys z-=ad
  else  += -= +=                       \   x+=xs y-=ys z+=ad
  then ;                               \ with: *ad=Atan(1>>i)

: trigo      \ --                      \ Trigo complete Rot.
  zt2  gw 1+ 0                         \ for i=0 to gw
  do   i (trigo)  2+     \  i trace \   TrigRot:zt[i]
  loop drop              \ gw trace \ --
;
\ x --> Kt*[x*cos(z) - y*sin(z)]   x --> Kt*sqr(x*x+y*y)
\ y --> Kt*[y*cos(z) + x*sin(z)]   y --> 0    {vectoring}
\ z --> 0          {rotating}      z --> z+Atan(y/x)

: (atg)        \ == zs+Atan(y/x)
  vectoring  y rw 0 skip nip           \ -- y<>0
  if   z zs :=  trigo
  then  0 z reg@  rad> ;

: fatan      \ f == Atan(f)            \ inverse trigo tangent
  y reg!  x 4 /4=  dup 0<              \ -- e e<0 \ y=m f=m<<e
  if    negate y                       \ if y<1: x=1    y=m>>-e
```

```
  else  x                            \ else    x=1>>e y=m
  then  dup >>=                      \ x and y aligned
  zs clear  (atg) ;                  \ == Atan(y/x)

: (s>c)      \ f ==                  \ exit:x=sqr(1-f^2) y=-f
  fdup0=
  if   fdrop  zs clear  y 4 /4=      \ if f=0: zs=0 y=1.
  else (?1-)  dup 1 and              \ -- f<0 E t/f \ zs=-|f|
       if   1-    z     >>           \ if E odd: E-=1 z/=2
       then (ath)                   \ x=Kh*sqr(1-f^2)<<(E/2+1)
       2/ 1+  1 z 1/Kh >>=           \ -- f<0 E/2+1 \ z=1/2Kh
       y=xz 1+  x y >>=              \ -- f<0  \ x=sqr(1-f^2)
       if   zs +/-
       then                          \ --      \ zs=-f
       y zs :=                       \ y=-f
  then ;
\ sqr[1-(1-m>>E)^2] = sqr[(2+m-m>>E)^2 -
\      (2-m-m>>E)^2]>>(E/2+1)

: fasin      \ f == Asin(f)          \ inverse trigo sine
  (s>c)  y +/-  zs clear  (atg) ;    \ == Atan[sqr(1-f^2)/f]

: facos      \ f == Acos(f)          \ inverse trigo cosine
  (s>c)  -1 zs ztrw >>=  (atg) ;     \ == pi/2-Asin(f)

: fs>c          \ f == sqr(1-f^2)    \ sine <--> cosine
  (s>c)  0 x reg@ ;

: (sct)         \ f == | -- Q        \ for sin, cos, tan
  >rad  1 ztrw (z/mod)               \ z=f-Q*pi/2 x=1/Kt y=0
  x 1/Kt :=  rotating trigo ;        \ x=cos(z) y=sin(z)
                                     \   Qmod4:  0  1  2  3
                                     \   cos(f)  x -y -x  y
                                     \   sin(f)  y  x -y -x

: (sc)          \ N -- | == f        \ sin:N=Q cos:N=Q+1
  >r 0  r@ 1 and                     \ -- 0 Nodd? | N
  if    x                            \ if Nmod4=1or3 use r=x
  else  y                            \ if Nmod4=2or4 use r=y
  then                               \ -- 0 r | N
  r> 2 and                           \ if Nmod4 = 2 or 3
  if    dup +/-                      \     negate r
  then  reg@ ;                       \ == f

: fsin          \ f == sin(f)        \ trigo sine
  (sct) (sc) ;

: fcos          \ f == cos(f)        \ trigo cosine
  (sct) 1+ (sc) ;                    \ cos(f+pi/2)=sin(f)

: ftan          \ f == tan(f)        \ trigo tangent
  (sct)  1 and                       \ f=z+Q*pi/2
  if   x y +=  y x -=  x y +=        \ if Q odd: (x,y)=(y,-x)
  then                               \ cos(f)=s*mx<<ex  s^2=1
  0 x norm  negate  y norm           \ sin(f)=s*my<<ey
  vectoring linear  1+ z reg@ ;      \ == (my/mx)<<(ey-ex)

\ trigo2 works like trigo, but where trigo uses 1/Kt
\ trigo2 uses zt1 table to repeat elementary rotations
\ such as to force the radius scaling factor to 2

: trigo2        \ --                 \ as trigo, force Kt=2
  zt1 zt2  gw 1+ 0                   \ for i=0 to gw
  do   over c@ 0                     \   repeat zt1[i] times
       do    j (trigo)       \  j trace \     TrigRot:zt[i]
       loop                          \      -- rt[i] zt[i]
       swap 1+ swap 2+               \    -- rt[i+1] zt[i+1]
  loop 2drop                \ gw trace \ --
;

: xy>rt         \ X Y == R T         \ Rectangular to Polar
  xy!  vectoring trigo2              \ 2R=|x<<e| T=z+(x<0)*pi
  -2 y ztrw >>=  minY                \ -- e xy<0 \ y=pi
  if  x +/-  z y  minZ               \ if x<0: correct x z
      if   -=                        \   if z>=0: z-=pi
      else +=                        \   if z<0 : z+=pi
      then                           \ z in [-pi,pi[ and x>0
  then                               \ -- e \ R=(x/2)<<e T=z
  1- x reg@  0 z reg@  rad> ;        \ (R,0) rot T = (X,Y)

: xyrot         \ X Y T == X' Y'     \ Rectangular Rotation
  >rad  2 ztrw (z/mod)               \ T=z+Q*pi
  rotating  xy!                      \ -- Q e \X=x<<e Y=y<<e
  trigo2  1- >r  1 and               \ (x,y) = 2*(x,y) rot z
  if  x +/-  y +/-                   \ if Q odd: negate x y
```

```
  then                              \ -- | e-1
  r@ x reg@  r> y reg@ ;            \ (X',Y') = (X,Y) rot T

: rt>xy          \ R T == X Y       \ Polar to Rectangular
  0 s>f fswap  xyrot ;              \ (X,Y) = (R,0) rot T

\ *****************************************
\              INPUT/OUTPUT
\ *****************************************

variable #places
         rw allot                   \ (0.5E-#places)>>5

: places         \ n --             \ set #places
  dup #places !
  negate s>f ln10 f* (exp) negate   \ -- e \ 10^-n = x>>e
  6 + #places 2+ x >>= ;            \ comma right of MS byte

6 places                            \ default

: dig-buff       \ -- addr ; where we build F.P. string
  here 64 +  ;

: ins-char       \ addr char -- addr+1
  over c!  1+  ;

: ins-dp         \ addr -- addr+1
  ascii . ins-char  ;

: ins-0          \ addr -- addr+1
  ascii 0 ins-char  ;

: -0trailing \ addr len -- addr len' ; remove trailing 0s
  dup 0
  ?do  2dup + 1- c@ ascii 0 <> ?leave  1-  loop  ;

: op-prep        \ f == | -- exp(10)
  bl dig-buff c!  fdup0=            \ assume sign is +ve
  if   fdrop  x clear  0            \ if f=0: -- 0 \ x=0
  else fsign                        \ else: use abs(f)
       if  fnegate  ascii - dig-buff c!  then
       fln ln10 f/mod               \ -- exp(10)
       (exp) negate                 \ -- e b \ f=s(x<<b)Ed
       5 + x dup >>=  x #places 2+ +=   \ -- e \ round x
       [ x rw + 2- ] literal  @     \ comma right of MS byte
       [ 10 256 * ] literal =
       if  1+                       \ if x=10 (after round)
           256 [ x rw + 2- ] literal !  \ -- e \ x=1
       then
  then  ;                           \ -- exp(10)

: m-digs         \ addr n -- addr+n \ convert n next digits
  0                                 \ (from x to addr)
  ?do  [ x rw + 2- ] literal @      \ pick up high word
       tuck  flip 15 and            \ extract high digit
       ascii 0 +  ins-char          \ make ascii, and stash
       swap  255 and                \ clear high digit
       [ x rw + 2- ] literal !      \ and multiply by 10
       -2 xs x >>=  x xs +=  x dup +=   \ (decimal shift)
  loop   ;

: (e.)  \ f == | -- addr len        \ format +d.ddddddE+dd
  op-prep                           \ form exponent
  dig-buff 1+  1 m-digs  ins-dp     \ 1st digit & d.p.
  #places @ m-digs                  \ next digits
  base @ >r  decimal                \ save base, set decimal
  swap dup                          \ -- addr exp exp
  abs 0                             \ abs value of exponent
  <# # #s  rot 0<                   \ min two digits
     if ascii - else ascii + then   \ with sign & E
     hold  ascii E hold             \ i.e. E+dd
  #> bounds                         \ copy characters
  do  i c@ ins-char  loop           \ incrementing address1
  dig-buff tuck -                   \ change to addr & len
  r> base !  ;                      \ restore base

: (f.)  \ f == | -- ad cnt          \ free format
  op-prep  dig-buff 1+              \ -- e addr
  over -3 #places @ between         \ display w/o exp?
  if  over 0<                       \ leading zeros?
      if  ins-0  ins-dp             \ leading zero & d.p.
          swap abs 1- 0             \ calc. zeros after d.p.
          ?do  ins-0  loop
          #places @ 1+ m-digs       \ significant digits
      else                          \ xxx.yy = x.xxyyE2
```

```
          over 1+ m-digs  ins-dp    \ 10..99 == E1
          #places @ rot - m-digs    \ digits after d.p.
      then                          \ -- addr
      dig-buff tuck - -0trailing    \ -- dig-buff len
  else                              \ display with exponent
      1 m-digs  ins-dp              \ first digit and d.p.
      #places @ m-digs              \ digits after d.p.
      dig-buff tuck - -0trailing +  \ remove any zeros
      swap dup  abs 0               \ abs value of exponent
      base @ >r  decimal            \ switch to decimal
      <#  #s  rot 0<                \ min 1 digit
          if ascii - hold  then     \ with sign if neg
          ascii e hold              \ i.e. e[-]d{d}
      #> bounds                     \ copy characters
      do  i c@ ins-char  loop       \ incrementing address
      dig-buff tuck -               \ change to addr & len
      r> base !                     \ restore base
  then  ;

: e.    \ f ==                      \ format with E +space
  (e.) type space  ;

: e.r   \ f == | n --               \ format with E in field
  (e.)  rot over - spaces type  ;

: f.    \ f ==                      \ format w/o E +space
  (f.) type space ;

: f.r   \ f == | n --               \ format w/o E in field
  (f.) rot over - spaces type ;

variable fpl   \ floating point location

: float?        \ -- fl ;
  fpl @ -32768 <> ;

: fnumber?      \ ad -- true | == f ; when float? true
                \ ad -- d fl | ==   ; when float? false
  dup 1+ c@ ascii - = dup >r -          \ -- ad | s
  -32768 fpl !  -1 dpl !  0 0 rot       \ -- ud ad | s
  begin  convert                        \ -- ud ad | s
         float? if dpl @ negate fpl +! then
         dup c@ 44 47 between           \ ascii ,-./
  while  dup c@ ascii . = if 0 fpl ! then
         0 dpl !
  repeat
  -rot r> if dnegate then rot           \ -- d ad
  dup c@ upc ascii E =                  \ -- d ad ?E
  if   dup 1+ c@ dup ascii - = >r ascii + = r@ or -
       0 dpl !  0 0 rot  convert        \ -- d ud ad
       nip swap r> if negate then       \ -- d ad n
       fpl float? if +! else ! then     \ -- d ad
  then
  c@ bl =
  dup float? and
  if   base @ 10 <> abort" base must be decimal"
       state @ abort" use F# in compilation"
       -rot d>f fpl @ ?dup if s>f falog f* then
  then ;

: fnumber       \ ad -- | == f
  fnumber? not ?missing ;

: (f#)  \ ad -- true | == f ;or     \ convert into float
        \ ad -- false               \ even if integer
  fnumber? dup float? not and if -rot d>f then ;

: f#in  \ -- 3 | == f ;or           \ input number, force
        \ -- 0                      \ conversion into float
  query bl word (f#) ;

: f#    \ == f ;exec
        \ ==   ;comp
  bl word         state @ >r state off \ hack the state
  (f#)                         r> state !    \ to avoid
#LITERAL
  not ?missing
  state @
  if   [compile] fliteral
  then ;
immediate

\ F# reads next number in tib, forces it to float
\ and returns it or compiles it (F# is state smart)
```

```
: f.s   \ ==                         \ dump the fp stack
  fp0@ fp @ - ?dup
  if    cr ." top" 0
        do  cr                       \ for each stack item:
            i (fpick) e.             \ print in E format
            fp @ i +                 \ -- itemAddr
            dup 2+ m.                \ print mantissa in hex
            ." <<" ?                 \ print binary exponent
        sw +loop
  else  cr ." Empty F.P. stack"
  then  cr ;

: fstatus       \ --                 \ display the prompt
  status? @ if base @ >r decimal     \ (for defer status)
  <# r@ 10 <>                        \ if base<>10
     if   r@  0 #s 2drop   ascii : hold
          \   display it after :
     then
     fp0@ fp @ - sw / 0 #s ascii . hold
     \ fp stack depth after .
     depth 2- ?dup                   \ if int.stack depth<>0
     if       0 #s 2drop   ascii - hold
              \   display it after -
     then                            \ -- d
  #> type space                      \ OK-int.float:base
  r> base ! then cr ;

also forth definitions               \ in forth dictionary

: floats        \ --
  ['] fnumber is number              \ enable FP input
  ['] fstatus is status ;            \ and float prompt

: integers      \ --
  ['] (number) is number             \ disable FP input
  ['] (status) is status ;           \ and normal prompt

previous definitions                 \ return into f-pack

EOF
\ ****************************************
\ IEEE 32 bits         (80287 32 bit fp)
\ ****************************************

\ bit31    = mantissa sign (negative:1  else:0)
\ bit30-23 = biaised binary exponent (bias = 128)
\ bit23-0  = mantissa normalized magnitude,
\            comma left of bit23=1, masked by exponent lsb

: f32array   \ dim -- ;comp     \ array of f32 variables
             \ index -- addr ;exec  \ to use with f32! f32@
  create  4 * allot
  does>   swap 4 * + ;

: f32!  \ ad -- | f ==               \ store 32bit float
  fdup0=
  if   4 erase                       \ zero: all zeroes
  else 0  fsign                      \ -- ad 0 f<0
       if  fnegate  drop   256       \ separate sign/modulus
       then                          \ then biaise exponent:
       x reg!   128 +                \ -- ad s be (f=*x<<e)
       dup   -256 and 2 ?ferror      \ check over/underflow
       +  x 24 round  5 x dup >>=    \ -- ad sbe (align lsb)
       128 *  [ x rw + 2- ] literal @ \ -- ad sbe hiw
       127 and or  over 2+ w!        \ make and store hi
       [ x rw + 4 - ] literal @ swap w! \ store lo
  then ;

: f32@  \ ad -- | == f               \ fetch 32bit float
  x clear
  dup w@ [ x rw + 4 - ] literal ! 2+ \ -- ad+2
      w@ 0 128 um/mod  swap 128 or \ -- sbe H \ unmask msb
         [ x rw + 2 - ] literal !
  dup  256 and if  x +/-  then       \ sign mantissa
  255 and [ 5 128 - ] literal +      \ unbiaise exponent
  x reg@ ;                           \ normalize and push

\ ****************************************
\ IEEE 64 bits         (80287 64 bit fp)
\ ****************************************

\ bit63    = mantissa sign (negative:1  else:0)
\ bit62-52 = biaised binary exponent (bias = 1024)
\ bit52-0  = mantissa normalized magnitude,
\            comma left of bit52=1, masked by exponent lsb

: f64array    \ dim -- ;comp     \ array of f64 variables
          \ index -- addr ;exec \ to use with f64! f64@
  create  8 * allot
  does>   swap 8 * + ;

: f64!  \ ad -- | f ==              \ store 32bit float
  fdup0=
  if   8 erase                      \ zero: all zeroes
  else 0  fsign                     \ -- ad 0 f<0
       if  fnegate  drop  2048      \ separate sign/modulu
       then                         \ then biaise exponent:
       x reg!  1024 +               \ -- ad s be (f=*x<<e)
       dup  -2048 and 2 ?ferror     \ check over/underflow
       +  x 53 round                \ -- ad sbe (align lsb)
       16  *  [ x rw + 2- ] literal @ \ -- ad sbe hiw
       15  and or  over 6 + w!      \ make and store hi
       [ x rw + 8 - ] literal @ over w! 2+ \ -- ad+2
       [ x rw + 6 - ] literal @ over w! 2+ \ -- ad+4
       [ x rw + 4 - ] literal @ swap w!    \ --
  then ;

: f64@  \ ad -- | == f              \ fetch 32bit float
  x clear
  dup w@ [ x rw + 8 - ] literal ! 2+    \ -- ad+2
  dup w@ [ x rw + 6 - ] literal ! 2+    \ -- ad+4
  dup w@ [ x rw + 4 - ] literal ! 2+    \ -- ad+6
      w@ 0  16 um/mod  swap   16 or
      \ -- sbe H \ unmask msb
         [ x rw + 2 - ] literal !
  dup 2048 and if  x +/-  then      \ sign mantissa
  2047 and 1024 -                   \ unbiaise exponent
  x reg@ ;                          \ normalize and push

\ ****************************************
\ PERFORMANCE MEASUREMENTS
\ repeat n times the next word in TIB
\ ****************************************

: monadic       \ f == f | n --     \ 1000 monadic fsqr
  ' swap  0 do                  \ for FSQR FLN FALN FLOG
    fdup  dup execute  fdrop       \     FSIN FCOS FTAN ...
  loop  drop  7 emit ;

: diadic        \ f2 f1 == f2 f1 | n -- \ 1000 diadic f*
  ' swap  0 do                      \ for F+ F- F* F/ F**
    fover fover  dup execute  fdrop
  loop  drop  7 emit ;

: complex  \ f2 f1 == f2 f1 | n -- \ 1000 complex xy>rt
  ' swap  0 do                      \ for XY>RT RT>XY
    fover fover dup execute fdrop fdrop
  loop  drop  7 emit ;

EOF
****************************************
```

CORDIC main functions and user interface

FP STACK

A separate stack is used for floats; this avoids mixed stack handling operators, and allows parallel integer and float computations. Integer and float stacks may communicate with:
S>F or D>F to convert single or double integer into float,
SINT or DINT to convert float into single or double integer (where NEAREST or TRUNCATE fix the rounding policy)

FLOAT INPUT FORMAT

Float number input must be in decimal; mantissa input is limited to 32 bits (9 decimal digits max) exponent input is limited to 15 bits (4 decimal digits max); (accuracy actually depends on SMB and on memory ops chosen).
Float number must have at least a "." or a "E" (or "e"):
when no "E", "." produces a float, "," produces a double
examples: 12.34   -123.   .123   1.234E3   -1234e+2   12,34E-15
Inside a definition, a float literal must be preceeded by F#; outside, F# may be avoided by executing FLOATS; INTEGERS undoes what FLOATS did (it restores deferred words).

!!! YOU MUST EXECUTE INTEGERS BEFORE FORGETTING CORDIC !!!

FLOAT OUTPUT
PLACES controls the number of mantissa digits to output (the maximum significant value you may give to PLACES is echoed while compiling CORDIC).
F. prints its argument in natural "free" format (12.3)
E. prints its argument in "exponential" format (1.230000E+01)
F.R and E.R print their argument right justified as does .R

FP FUNCTIONS NAMES
Most fp functions have their integer equivalent; their name is built by prefixing their integer equivalent with F
(FDUP FSWAP FROT ... F+ F- F* F/ ... F. F.S FSTATUS ...)

TRIGONOMETRIC UNITS
Trigonometric functions have built in deferred conversions, allowing their arguments (or result) to be expressed in RADIANS DEGREES or GRADES (3.14rad = 180deg = 200grd); (execute theese words to set the angle unit you wish).
RAD>DEG DEG>RAD RAD>GRD GRD>RAD do explicit conversions.

LISTING: CORDIC.VOC

```
**********    CORDIC VOCABULARY GLOSSARY    **********
     Floating Point package with sizable accuracy
     STACK NOTATIONS:
          (==f) denotes float stack effect
          (n--|==f) denotes integer and float stacks
          effects
     FP STACK HANDLING
          FDROP FNIP FDUP FOVER FPICK FSWAP FROT FROLL
          F.S FSTATUS
     LOGIC
          FDUP0= FSIGN F0< F0= F0> F< F= F> ?FERROR
     INTEGER AND STRING CONVERSIONS
          D>F S>F DINT SINT NEAREST TRUNCATE
          PLACES F. F.R E. E.R
          FLOATS INTEGERS FLOAT? FNUMBER? FNUMBER F# F#IN
     ARITHMETIC
          FNEGATE FABS FMAX FMIN F+ F- F* F/ F1/ F/MOD
          FFRAC FINT
     HYPERBOLIC
          FSQR FLN FALN FLOG FALOG F**
          FSINH FCOSH FTANH FATANH
     TRIGONOMETRIC
          FSIN FCOS FTAN FASIN FACOS FATAN FS>C RT>XY XY>RT
          XYROT
          RAD>DEG DEGREES DEG>RAD RADIANS RAD>GRD GRADES
          GRD>RAD
     MEMORY
          F@ F! F32@ F32! F64@ F64!
          F, FLIT FLITERAL FCONSTANT FVARIABLE F*ARRAY
```

?FERROR (b,n--) if b=TRUE, give the nth fp error :

0	empty FP stack	concern most fp words
1	full FP stack	concern most fp words
2	conversion overflow	SINT DINT F32! F64!
3	overflow	concern most fp words
4	underflow	concern most fp words
5	divide: arg=0	F/ F1/ F/MOD FFRAC
6	logarithm: arg=0	FLN FLOG F**
7	logarithm: arg<0	FLN FLOG F**
8	square root: arg<0	FSQR
9	asin/acos/atanh: |arg|>1	FS>C FASIN FACOS FATANH

D>F (d--|==f) convert double (from integer stack) to float (on fp stack) "conversion overflow" error if float too big for a double

DEG>RAD (f==f') f in degrees is converted into f' in radians (180deg=3.14rad)

DEGREES (--) set Degree mode for all trigonometric functions

DINT (f==|--d) convert float (from fp stack) to double (on integer stack)

E. (f==) print f in exponential format, ex: 3.141593E+00 with as many fractionals digits as was set using PLACES

E.R (n--|f==) print f in exponential format (see E.) right justified in a field of n characters (as .R does)

F! (ad--|f==) store f as is on the fp stack, at address ad

F# exec:(==f) convert next word in input stream into a float comp:(==) if compiling, compile it as a float literal "is undefined" error if unable to convert

F#IN (==f|--TRUE) query text (until cr) and convert it to a float
(==|--FALSE) if conversion fails, push only FALSE flag

F* (f2,f1==f) f=f2*f1 multiply fp stack second by top

F** (f2,f1==f) f=f2^f1=exp(f1*ln(f2)) raise f2 to the power f1
"logarithm: arg=0" error if f2=0, "logarithm: arg<0" if f2<0

F*ARRAY (wn...w1,n <name> -- ; in...i1--ad)
compiles multidimensionnal arrays of floats (in stack format); when compiling, pop n, then n array dimensions (wi>1), and allot space for wn*...w2*w1 floats (be carefull with memory!); when executing, pop n array indexes (each ii should be 0<=ii<wi), and return the address of the corresponding float (for F@ or F!)

F+ (f2,f1==f) f=f2+f1 add fp stack top to second

F, (f==) compile f as a float literal

F- (f2,f1==f) f=f2-f1 subtract fp stack top from second

F. (f==) print f in free format, ex: 3.141593 0.5 1000. with a maximum of n+1 digits for n PLACES (n=6 in examples)

F.R (n--|f==) print f in free format (as F.), right justified in a field of n characters (as .R)

F.S (==) dump fp stack (as .S dumps integer stack)
each item is printed in exponential format and in internal format (full mantissa in hex, << , and signed binary exponent in decimal)

F/ (f2,f1==f) f=f2/f1 divide fp stack second by top
"FP divide: arg=0" error if f1=0

F/MOD (f2,f1==r|--q) f2=f1*q+r return integer quotient and float remainder of f2/f1; r sign = f1 sign (as for /MOD) "FP divide: arg=0" error if f1=0

F0< (f==|--b) b=TRUE if f negative, else b=FALSE

F0= (f==|--b) b=TRUE if f null (f modulus less than 2^-16384 = 10^-4932)

F0> (f==|--b) b=TRUE if f positive, else b=FALSE

F1/ (f==f') f'=1/f reciprocal of fp stack top
"FP divide: arg=0" error if f=0

F32! (ad--|f==) store f as a 32bits IEEE float, at address ad

F32@ (ad--|==f) fetch f as a 32bits IEEE float, from address ad

F64! (ad--|f==) store f as a 64bits IEEE float, at address ad

F64@ (ad--|==f) fetch f as a 64bits IEEE float, from address ad

F< (f2,f1==|--b) b=TRUE if f2 is less than f1, else b=FALSE

F= (f2,f1==|--b) b=TRUE if f2=f1 (|f2-f1| < 2^-16384 = 10^-4932)

F> (f2,f1==|--b) b=TRUE if f2 is greater than f1, else b=FALSE

F@ (ad--|==f) fetch f (in stack format) from adress ad

FABS (f==f') f'=|f| negate fp stack top if negative

FACOS (f==f') f'=Acos(f) <=> f=cos(f') inverse trigonometric cosine
return f' in the current angle unit (ex: DEGREES) in range [0,180] "asin/acos/atanh: |arg|>1" error if f modulus greater than 1

FALN (f==f') f'=exp(f)=e^f <=> f=ln(f') natural exponentiation

FALOG (f==f') f'=10^f <=> f=log(f') decimal exponentiation

FASIN (f==f') f'=Asin(f) <=> f=sin(f') inverse trigonometric sine return f' in the current angle unit (ex: DEGREES) in range [-90,90] "asin/acos/atanh: |arg|>1" error if f modulus greater than 1

FATAN (f==f') f'=Atan(f) <=> f=tan(f') inverse trigonometric tangent return f' in the current angle unit (ex: DEGREES) in range [-90,90]

FATANH (f==f') f'=Atanh(f) <=> f=tanh(f') inverse hyperbolic tangent "asin/acos/atanh: |arg|>1" error if f modulus greater than 1

FCONSTANT (f <name> == ; ==f) compile a float constant

FCOS (f==f') f'=cos(f) trigonometric cosine
takes f in the current unit (RADIANS DEGREES or GRADES)

FCOSH (f==f') f'=cosh(f)=(exp(f)+exp(-f))/2 hyperbolic cosine

FDROP (f==) drop fp stack top

FDUP (f==f,f) duplicate fp stack top

FDUP0= (f==|--b) b=TRUE if f null (f modulus less than 2^-16384 = 10^-4932)

FFRAC (f2,f1==f) fractional remainder of f2/f1; f sign = f1 sign (as /MOD) "FP divide: arg=0" error if f1=0

FINT (f==f') float integer part of stack top

FLIT (==f) word compiled by FLITERAL (as LIT is compiled by LITERAL)
DON'T TRY TO INTERPRET IT

FLITERAL (f==) compile a float literal (as LITERAL compiles an integer)
IMMEDIATE : USE ONLY IN COMPILATION MODE

FLN (f==f') f'=ln(f) natural logarithm (ln(e)=1)
"logarithm: arg=0" error if f=0; "logarithm: arg<0" error if f<0

FLOAT? (--b) b=TRUE if last number input was a float, else b=FALSE

FLOATS (==) allows direct input of float literals (without F#) F# must still be used in compilation mode ( ] should be modified)
INTEGERS must be executed BEFORE FORGETTING CORDIC VOCABULARY

FLOG (f==f') f'=log(f) decimal logarithm (log(10)=1)
"logarithm: arg=0" error if f=0; "logarithm: arg<0" error if f<0

FMAX (f2,f1==f) f=Max(f1,f2) return greater of f1 and f2

FMIN (f2,f1==f) f=Min(f1,f2) return lesser of f1 and f2

FNEGATE (f==f') f'=-f negate fp stack top

FNUMBER convert next word in input stream into a double or float
if FLOAT? returns TRUE (==f), else (--d); "is undefined" error
if conversion failed

FNUMBER? convert next word in input stream into a double or float
if conversion failed (--FALSE), else :
if FLOAT? returns TRUE (==f|--TRUE), else (--d,TRUE)

FOVER (f2,f1==f2,f1,f2) duplicate fp stack second

FPICK (n--|fn,...,f0==fn,...,f0,fn) duplicate the (n+1)th fp stack entry

FROLL (n--|fn,fn-1,...,f0==fn-1,...,f0,fn) rotate n+1 fp stack tops

FROT (f3,f2,f1==f2,f1,f3) rotate fp stack top, second and third

FS>C (f==f') f'=sqr(1-f*f) <=> [f=sin(a) => f'=cos(a)] "asin/acos/atanh: |arg|>1" error if f modulus greater than 1

FSIGN (f==f|--b) b=TRUE if f negative, else b=FALSE

FSIN (f==f') f'=sin(f) trigonometric sine
takes f in the current angle unit (RADIANS DEGREES or GRADES)

FSINH (f==f') f'=sinh(f)=(exp(f)-exp(-f))/2 hyperbolic sine

FSQR (f==f') f'=sqr(f) <=> f=f'*f' square root
"FSQR arg<0" error if f<0

FSWAP (f2,f1==f1,f2) swap fp stack top and second

FTAN (f==f') f'=tan(f)=sin(f)/cos(f) trigonometric tangent; takes f in the current angle unit (RADIANS DEGREES or GRADES)

FTANH (f==f') f'=tanh(f)=sinh(f)/cosh(f) hyperbolic tangent

FVARIABLE (<name> -- ; --ad) compile a float variable, for use by F@ or F!

GRADES (--) set Grade mode for all trigonometric functions

GRD>RAD (f==f') f in grades is converted into f' in radians (200grd=3.14rad)

INTEGERS (--) undo what FLOATS did

NEAREST (--) set rounding policy for float-to-integer conversions (0.5 is added to a float before its fractional part is truncated)

PI (==3.14...) push the famous constant onto the fp stack

PLACES (n--) set the number of digits to display by the words F. F.R E. E.R

RAD>DEG (f==f') f in radians is converted into f' in degrees (180deg=3.14rad)

RAD>GRD (f==f') f in radians is converted into f' in grades (200grd=3.14rad)

RADIANS (--) set Radian mode for all trigonometric functions

RT>XY (R,A==x,y) convert a complex number from polar (Radius,Angle) to rectangular (x,y) form : x=R*cos(A), y=R*sin(A). A is taken in the current angle unit (RADIANS DEGREES or GRADES)

S>F (n--|==f) convert single (from integer stack) to float (on fp stack)

SINT (f==|--n) convert float (from fp stack) to single (on integer stack); "conversion overflow" error if float too big for a single

TRUNCATE (--) set rounding policy for float-to-integer conversions (truncation is also called "rounding to minus infinity")

XY>RT (x,y==R,A) convert a complex number from rectangular (x,y) to polar (Radius,Angle) form : R=sqr(x*x+y*y), A=atan(y/x) in range [-PI,PI]
A is returned in the current angle unit (RADIANS DEGREES or GRADES)

XYROT (x,y,A==x',y') rotate a complex number (x,y) by the angle A; x'=x*cos(A)-y*sin(A), y'=x*sin(A)+y*cos(A); A is taken in the current angle unit (RADIANS DEGREES or GRADES)

■

TELEMATIQUE

CONTENU DU FORUM SAM*JEDI

SECRETAIRE Du 10.02.89 A 09h46
NOUVEAUX SOFTS EN TELECHARGEMENT:
EN FORTH:
1) UTILMENU.FTH menus deroulants
3) PI.FTH calcul de PI jusqu'a 22000 decimales. Un regal de matheux.
SYRACUSE.FTH les suites de SYRACUSE, se referer a l'article a paraitre dans JEDI 49 (livre dans une semaine).
4) DOSEDIT.FTH editeur de commandes DOS applique au tampon d'entree TURBO-FORTH.

EN C:
XRECV.C reception xmodem
XTRANS.C emission xmodem

DIVERS:
QUICKCRT.COM affichage ecran plus rapide. Efficace sur PC XT a 4,77 Mhz d'horloge.
MAPMEM.COM carte memoire ecran. Permet de connaitre la position memoire des progs residents.
SFIND.COM recherche de fichier a travers divers repertoire. Utile si vous avez beaucoup de sous-repertoires sur disque dur.

SECRETAIRE Du 10.02.89 A 09h55

A PARTIR D'AUJOURD'HUI VIENT D'ETRE MIS en place la routine de telemaintenance des BALs. Si vous avez perdu votre code d'acces, contactez-moi en laissant vos coordonnees dans ma bal ou en m'appelant au 1-49856367 hdb.

Le choix EXPEDITION COURRIER AU SYSOP a egalement ete active. Tout courrier expedie au SYSOP tombe dans ma BAL. Donc, ne cherchez plus mon no de BAL dans l'annuaire pour m'ecrire, vous gagnerez du temps si vous ne vous adressez qu'a moi-meme.

SECRETAIRE Du 10.02.89 A 10h03
QUELQUES MOTS SUR LES ACTIVITES TELEMATIQUES DE JEDI: le trafic 3615 SAM*JEDI est en progression reguliere et atteint une moyenne journaliere de 120 minutes.
C'est encore insuffisant pour amortir l'abonnement TRANSPAC, mais c'est encourageant. Je n'ecris pas ca pour vous pousser a consommer du 3615, mais plutot pour vous feliciter de votre fidelite et de la correction qui regne sur ce serveur.
Dans l'ensemble, le niveau des messages contenus dans le FORUM est tres eleve, meme si certaines semaines j'ai un peu l'impression d'etre seul.
Je rappelle encore une fois que ce forum vous permet de poser toutes les questions sur le langage FORTH pour les versions suivantes:
- TURBO-Forth PC
- FORTH LAXEN ET PERRY MSDOS
- FORTH LAXEN ET PERRY CP/M (Amstrad)
- un peu FORTH LMI
- NOVIX NC4016 et HARRIS RTX 2000 (qui disposera prochainement d'une sous-rubrique dans le telechargement FORTH).

Plus que sur dautres serveurs, nous nous contraignons a repondre le plus completement possible et dans des delais raisonnables a toutes vos questions.

Bien entendu, si vous-meme voulez repondre a une question posee sur le forum, vous pouvez le faire. Parfois plusieurs avis valent mieux qu'un seul.

Et pour finir, nous reclamons votre indulgence si nous ne pouvons vous satisfaire completement; nous sommes d'abord une association de benevoles et nous ne nous consacrons pas a plein temps a l'animation du serveur. Donc, pardonnez les cafouillages qui peuvent encore subsister de ci, de la, dans SAM*JEDI.

FORTH7 Du 11.02.89 A 15h27
BOUQUINS: les Quick Reference Guides de Microsoft Press sont de vraies mines pour les programmeurs: PSI a la bonne idee de les traduire en francais.

Saviez-vous que Ray DUNCAN, celui de BYTE, de "MS-DOS Avance" et de ces guides travaille en FORTH ?

Ray a declare a la derniere '88 Forth Convention que FORTH lui a fait gagner 6 a 8 mois sur d'autres auteurs pour sortir un guide OS/2 chez Microsoft Press! Il n'a pas dit combien de dollars...

Le 19 Novembre 1968, le premier FORTH disait 'OK' sur une console d'IBM 1130. 20 ans apres Charles MOORE, Chuck pour

les intimes, a toujours la frite: Forth doit continuer d'evoluer sans trop se preoccuper des efforts de standardisation car c'est sa grande force sur tous les autres langages! Et de nous livrer une reflexion digne du pere du Forth:
- la documentation d'un programme devrait pouvoir etre redigee en parallele avec la programmation selon des techniques interactives similaires...
Duncan, Moore: on en revient toujours a la 'DOC' veritable nerf de la guerre! (=Nouvel appel pour des doc-redacteurs de TURBO-FORTH...)

DANELUZZO Du 12.02.89 A 15h28
LE SYSTEME DE DEVELOPPEMENT DU RTX2000 EST RELIE AU PC PAR LIAISON SERIE. N'EST-IL PAS PLUS INTERESSANT D'AVOIR UN RTX SUR LE BUS ? POUR ACCES PAR EX A UNE CARTE DE SAISIES NUMERIQUE DEJA SUR LE BUS.

IAPX Du 12.02.89 A 19h58
KERMITPC.C S'EST CORRECTEMENT CHARGE EN ASCII SUR PC AVEC CARTE MODEM PNB. A LA COMPILATION AVEC TURBO C BORLAND V1.5 BEAUCOUP DE PROBLEMES MAINTENANT RESOLUS. MANQUE SEULEMENT LE FICHIER HEADER "SGTTY.H" --- COMMENT SE LE PROCURER? EST-IL DISPONIBLE SUR CE SERVEUR? IL SERAIT DOMMAGE DE PERDRE, OUTRE LE TELECHARGEMENT, LE TEMPS DEJA PASSE A L'ADAPTATION A TURBO C. UNE REPONSE RAPIDE M'OBLIGERAIT... MERCI

SECRETAIRE Du 13.02.89 A 09h50
REPONSE A DANELUZZO, sujet RTXDS HARRIS: Oui, a notre sens, le systeme de developpement HARRIS par liaison serie est discutable. Mais HARRIS a surtout tenu a ne pas privilegier le developpement sur PC ou compatible.
Pour repondre a votre demande, il existe une solution, la carte SC/FOX equipee du HARRIS RTX 2000 et qui est l'equivalent de la carte NB4100 de NOVIX:
- memoire integralement accessible par le PC.
- travail en // avec PC hote.
- inclus un compilateur FORTH.
Livraison en deux semaines, prix unitaire 1995$, accompte a la commande 200$ a verser a:
SILICON COMPOSER INC.
210 California Ave., Suite K,
Palo Alto CA 94306 (USA)
Tel: 19-1-415 322-8763 (depuis la France).

Une remarque personnelle en passant: le prix d'une telle carte nous semble eleve et si le HARRIS RTX 2000 est aussi prometteur qu'il y parrait, il serait etonnant qu'un industriel (ou plusieurs), ne sorte(nt) pas un produit

equivalent a moindre cout.

Alors petits JEDIs malins, si vous etes bricoleurs, derriere JOBS et WOZNIAK, il y a encore deux places a prendre sur la banquette arriere de la legende des precurseurs de la micro-informatique.

SECRETAIRE Du 13.02.89 A 10h13
REPONSE A IAPX: Desole, mais nous n'avons pas SGTTY.H dans nos disquettes. Le logiciel KERMITPC a ete ecrit initialement pour C MICROSOFT (il semble...) et naturellement, SGTTY.H ne figure pas dans le package TURBOC de Borland. Nous prenons note de vos deboires et eviteront desormais de mettre en telechargement des logiciels necessitant l'exploitation de modules non disponibles sur le serveur.
Mais si vous reussissez a adapter le fichier KERMITPC a TURBOC, nous sommes preneurs pour le mettre en telechargement.

DANELUZZO Du 13.02.89 A 12h57
UNE CARTE A RTX2000 EST DISPONIBLE PAR UNE AUTRE BOITE.
SMIS
130 ELM STREET
CAMBRIDGE MA 02139
617 354 7541
ELLE EST LIEE A DES CARTES D'AQUISITION DE DONNEES OU IMAGES. DISPOSE DE 'C' FORTH ET ASSEMBLEUR. JE NE SAIS PAS SI ELLE VAUT MOINS DE 1990$ ENCORE

IAPX Du 13.02.89 A 22h25
REPONSE A : REPONSE DE "SECRETAIRE" A IAPX LE 13.02.89
- VOUS TERMINEZ VOTRE REPONSE PAR: "AVEC VOTRE INDULGENCE..."
VOUS N'AVEZ AUCUNEMENT BESOIN D'INDULGENCE: JE TIENS A VOUS DIRE: BRAVO. EN EFFET, VOUS DITES VOUS MEME DANS UN DES MESSAGES DU FORUM "NOUS SOMMES DES BENEVOLES".
POUR AVOIR ESSAYE UN NOMBRE CERTAIN DE SERVEURS INFORMATIQUES, JE CONSTATE QUE DES BENEVOLES PRODUISENT UNE QUALITE APPAREMENT HORS DE PORTEE DES "PROFESSIONNELS".
2-REPONSE NON DILATOIRE --) ELLE EST TECHNIQUEMENT VALABLE - C'EST RARE SUR MINITEL
3-VOUS N'ETES PAS RESPONSABLE DES FICHIERS HEADER (.H) CONTENUS DANS LES SOURCES --) TOUT AU PLUS POURRIEZ VOUS INDIQUER, DANS LES INFORMATIONS DE TELECHARGEMENT, LE COMPILATEUR QUI DOIT ETRE UTILISE (MARQUE, VERSION).

SECRETAIRE Du 14.02.89 A 09h48
A IAPX ET AUTRES FANAS DU TELECHARGEMENT: nous privilegions avant tout le langage FORTH et plus particulierement TURBO-FORTH qui est notre creation. Les autres logiciels disponibles en telechargement ne sont pas de notre cru, donc sans garantie. Notre logiciel de gestion des fichiers TELECHARGEMENT etant malheureusement tres restrictif, il n'est pas possible de documenter plus aisement un programme. On nous a deja fait cette remarque, et conscients des desagrements que causent un logiciel "paye" mais inexploitable, on etudie un service GUIDE associe a chaque titre de la logitheque qui resumera en une page les contraintes d'exploitation des logiciels.

Notre but n'est pas de vous faire stationner dans un serveur "tortue", mais de renouer avec la tradition des RBBS americains ou l'on echange des infos et des programmes (publics domain et freeware). Notre satisfaction est plus grande de savoir que vous revenez regulierement pour nous consulter. Ici nous sommes entre copains devant le feu de cheminee et non derriere un tiroir caisse.

D'ailleurs un conseil, pour ne pas rester trop longtemps branche sur SAM*JEDI, equipez-vous d'un cable MINITEL PC ou d'une carte MODEM et d'un logiciel de capture ASCII (PROCOMM, LCECOM, etc...) et revoyez a tete reposee nos messages sans alourdir votre budget. Mais en ce qui concerne IAPX c'est deja le cas je pense...

ESTHETE Du 14.02.89 A 20h54
VF83 NE COMPORTE PAS DE VOCS. DONC JE REGARDE DANS TF83... ET LA JE NE TROUVE PAS BODY> ?
JE COULE..... MERCI BYE

SECRETAIRE Du 15.02.89 A 11h00
REPONSE A ESTHETE CONCERNANT BODY>. CE MOT PERMET TRES CERTAINEMENT UNE TRANSFORMATION DU TYPE CFA EN PFA OU INVERSEMENT. SUR LA PLUPART DES FORTHs, LE PFA EST SITUE IMMEDIATEMENT APRES LE CFA LEQUEL EST OBTENU PAR LE MOT ' (PRONONCER TICK); EX:
' DUP DELIVRE LE CFA DE DUP
ENSUITE, >BODY TRANSFORME CE CFA EN PFA QUI EN TURBO-F83 CORRESPOND A CFA+2. JE DEDUIS QUE BODY> SUR VF83 FAIT L'INVERSE.

SECRETAIRE Du 15.02.89 A 11h08
euroFORML CETTE ANNEE EST ORGANISE PAR LE FIG HAMBOURG ET SE PASSERA DU 13 AU 15 OCTOBRE 1989 A L'HOTEL SELAU PRES DE NUREMBERG (RFA). POUR PRECISIONS, CONTACTER:
M.KERN c/o DELTA GmbH
Roter Hahn 42
D - 2000 HAMBURG 72 (RFA/BRD)
OU TEL:
19-49 40 644 5782 DEPUIS LA FRANCE.
THEME DE CETTE ANNEE: FORTH DANS LES APPLICATIONS EN TEMPS REEL ET SES POTENTIALITES EN ROBOTIQUE ET CONTROLE IDUST.

SECRETAIRE Du 17.02.89 A 13h19
POUR DEBUTANTS EN FORTH: conversion degres fahrenheit en celsius et inversement:

```
: C>F  9 5 */ 32 + ;
: F>C  32 - 5 9 */ ;
```

Exemple:

```
: .TREMPE  ( celsius ---)
DUP 6 .R ." C  " C>F 6 .R ." F" ;
: TREMPE-ACIER ( ---)
CR ." ROUGE FONCE       " 680 .TREMPE
CR ." ROUGE CLAIR       " 850 .TREMPE
CR ." ROUGE TRES CLAIR "
```

SECRETAIRE Du 17.02.89 A 17h22
DES GRAPHES EN 2 COUPS DE CUILLER A POT:

```
80 STRING ANNEE$
" JAN FEV MAR AVR MAI JUN " ANNEE$ $!
" JUL AOU SEP OCT NOV DEC " ANNEE$
APPEND$
: GRAPHE ( n12 n11..n1 ---)
DEPTH 12 < ABORT" Manque de donnees"
DARK 13 1 DO ANNEE$ I ITEM TYPE
SPACE 220 REPLICATE CR LOOP 3 SPACES
192 EMIT 72 1
DO I 5 MOD 0= IF 193 EMIT ELSE 196 EMIT
THEN LOOP  CR 2 SPACES
71 0 DO I 0 <# # # #> TYPE 3 SPACES
5 +LOOP ;
```

Exemple d'utilisation:

```
: DATAS88 10 30 25 40 50 12 17 2
10 70 40 30 ;   DATAS88 GRAPHE
```

LAMBERTPH Du 18.02.89 A 01h30
DE LAMBERT PH - CARTE MODEM KORTEX 1200 ENFIN DANS MON PC - MAIS POUR LE TELECHARGEMENT? - QUELQU'UN SAIT-IL QUELLE PROCEDURE SUIVRE AVEC KXCOM2 TRANSFERER EN KERMIT? LA DOC EST UN PEU LEGERE A CE SUJET - ET C'EST FRUSTRANT - MERCI D'AVANCE -

---

JLUC Du 18.02.89 A 16h59
Quelques questions a propos de turbo-forth.
1) Au lancement (dans ma version), il prend le contenu de l'adresse 6 pour initialiser RP, ce qui donne FEFO comme depart de la pile de retour. Pourquoi, cette adresse 6? Elle se trouve dans le PSP d'un .COM et pour MSDOS elle correspond a un saut a l'interruption 21h CALL F01D:FEF0
Est-ce un reste de compatibilite avec CP/M.? Peut-on utiliser la partie superieure de FEF0 a FFFF sans danger?
2) Je manque de place pour une de mes applications: je compile aux alentours de D800 apres avoir reduit le noyau forth a sa moitie et mis FILES a 3. Pour garder une facilite d'edition je ne voudrais pas a chaque fois metacompiler meme sans entete.
Pour le moment j'aimerais enlever le VIEW-FIELD qui me permettrait de recuperer 4000 octets. Dans KERNEL, j'ai rajoute devant VOCABULARY FORTH un 0 ,-T. J'ai reecrit "CREATE avec COUNT HERE 2+ PLACE HERE 0 , etc
Dans le META j'ai enleve le 0 ,-T du mot HEADER et ca ne fonctionne pas correctement.
3) (qui n'engage que moi). Le mot FILENAME ne pourrait-il pas etre vectorise pour ne pas etre oblige de reecrire OPEN SAVE LSAVE etc quand le nom de fichier n'est pas dans le flot d'entree mais a une adresse quelconque en memoire.
On pourrait faire:
7 STRING ESS$ 0 C, " ESS.FTH" ESS$ $!

: ESS2 ESS$ DROP ;
et ' ESS2 IS FILENAME
D'autres mots comme] INTERPRET RUN gagneraient a etre vectorises. NUMBER l'est bien lui !!!
4) Suite logique au 3). Les chaines de caracteres ne pourraient-elles pas avoir systematiquement un 0 (ou $) au bout qui se deplacerait en meme temps que la chaine s'allonge. On gagnerait une compatibilite avec MSDOS
M'ENFIN turbo tel quel est deja pas mal!

Au fait personne n'essaie d'ecrire un turbo.EXE !! (dur dur)

IAPX Du 20.02.89 A 00h20
SUGGESTION: UN PROGRAMME CHARGEABLE ET EXPLOITABLE SUR TOUT PC ET COMPATIBLE:
ECRIT EN GWBASIC ET SAUVE EN MODE ASCII, IL SERAIT "CAPTURABLE" PAR TOUT SITE EQUIPE D'UNE CARTE MODEM OU D'UN CORDON MINITEL-PC, CE PROGRAMME CREERAIT - SOIT PAR DES POKES DE "DATA" PUIS SAUVEGARDE D'UNE ZONE MEMOIRE - SOIT PAR CREATION DIRECTE D'UN FICHIER DISQUE PAR DES "OUTPUT" - UN MODULE DE TELECHARGEMENT (CE POURRAIT ETRE KERMIT-PC DISPONIBLE SOUS CETTE FORME).
QU'EN PENSE NOTRE SECRETAIRE ?

SECRETAIRE Du 20.02.89 A 09h39
On peut resoudre le probleme de IAPX par l'utilisation de DEBUG. Creer un fichier ASCII par COPY CON: CLSEGA.TXT en y mettant:

```
E 100 B8 12 11 B3 00 CD 10 29
E 108 C0 8E D8 FF 36 87 04 80
E 110 0E 87 04 01 B9 00 06 B4
E 118 01 CD 10 8F 06 87 04 BA
E 120 B4 03 B8 14 07 EF B8 00
E 128 06 B7 71 31 C9 BA 4F 2A
E 130 CD 10 B4 02 30 FF 31 D2
E 138 CD 10 CD 20 74 03 C6 07
E 140 00 A0 67 46 32 06 69 46
E 148 22 C5 22 C1 74 03 C6 07
N CLSEGA.COM
RCX
50
W
Q
^Z
```

Et de retour sous DOS:
DEBUG < CLSEGA.TXT
Ainsi DEBUG "digere" les commandes de CLSEGA.TXT et fabrique un CLSEGA.COM.
MAintenant, si vous voulez desassembler la merveille ainsi assemblee, il suffit de taper:

```
COPY CON: DESASS.TXT
 U 100,150
 Q
 ^Z
```

Et sous DOS, de taper:
DEBUG CLSEGA.COM < DESASS.TXT > CLSEGA.ASM
Et voila; a noter que DEBUG.COM accepte aussi des mnemoniques assembleur, ce qui peut donner des idees du genre:

```
COPY CON: EGA25.ASM
N EGA25.COM
A 100
MOV  AX,1111
MOV  BL,00
INT  10
SUB  AX,AX
MOV  DS,AX
PUSH [0478]
OR   BYTE PTR [0487],01
MOV  CX,0B0D
MOV  AH,01
INT  10
POP  [0487]
MOV  DX,03B4
MOV  AX,0D14
OUT  DX,AX
INT  20
RCX
28
W
Q ^Z
```

Puis sous DOS: DEBUG < EGA25.ASM

SECRETAIRE Du 21.02.89 A 17h52
NOUVEAUX SOFTS EN TELECHARGEMENT:
C: - FASTWR.C ACCES DIRECT VIDEO
- VIDASM.C CTRL VIDEO MODE
- QLIB.C TRANSFORMATION DE .OBJ EN .LIB
FORTH, NOUVELLE RUBRIQUE NOVIX NC4016 ET HARRIS RTX 2000:
- NOYAUNX.FTH ASEMBLEUR NOVIX POUR TURBO.FORTH; GENERE DU CODE NOVIX.
- MAIN.FTH GENERATEUR NOYAU NC4016
- COM.FTH INTERFACE COMMUNICATION ENTRE TURBO-FORTH ET CARTE NB4100
- FDISQUE.FTH POUR FAIRE LIRE DSK DOS AU NC4016
- FCONSOLE.FTH POUR GERER CLAVIER ET ECRAN
- FUNCT.FTH GESTION HEURE SYSTEME
- INITFUNC.FTH INITIALISATION DES FONCTIONS D'INTERFACE NC4016 AVEC TF83
- NOVIX.FTH CONTROLE DES FONCTIONS ENTRE NOVIX NC4016 ET PC.
TOUS CES FICHIERS SONT ENCORE DES PROTOTYPES ET DES RECEPTION DE VERSIONS AMELIOREES, NOUS LES REMPLACERONS.
POUR CEUX QUI NE CONNAISSENT PAS LE NOVIX NC4016, ILS POURRONT SE REPORTER A L'ARTICLE QUE J'AI ECRIS DANS MICRO-SYSTEME DE MAI 1988. BONNE SOIREE LES COUCHE-TARD...

LAMBERTPH Du 22.02.89 A 23h16
DE LAMBERT PH A JLUC: BEAUCOUP TROP DE QUESTIONS POUR CE SOIR. POUR CE QUI EST DE L'ADRESSE 6 DU PSP, IL EST PRECISE DANS LE TECH-REF IBM QUE "THE WORD AT OFFSET 6 CONTAINS THE NUMBER OF BYTES AVIABLE IN THE SEGMENT". QUELQUES TEST SEMBLE DEMONTRER QUE CE MOT EST MIS A JOUR LORS DES CHARGEMENTS DES .COM ET PROBABLEMENT LORS DES RESERVATIONS D'ESPACE. (UTILISATION DE TURBO FORTH SUR 32 KO).

SECRETAIRE Du 23.02.89 A 13h52
SIMULER DIM T$(10) DU BASIC EN TF83: Pour definir un tableau de chaine utilisable comme un tableau dimensionne de basic, definir:

```
  40 STRING T$(1)
  40 STRING T$(2)
  40 STRING T$(3)...etc... jusqu'a
  ...40 STRING T$(10)
  11 CASE: T$() NOOP
 T$(1) T$(2) T$(3)...etc... T$(10) ;
```

et s'utilise ensuite sous la forme:
" Coefficient de striction:" T$(1) $!
" Moment forces angulaires:" T$(2) $!
et pour afficher une chaine depuis le tableau T$(), taper:
n T$() TYPE
Exemple: 2 T$() TYPE
affiche "Moment forces angulaires:"

TICK Du 23.02.89 A 17h53
SAUT LES EMPILES, JE MARCHE A L'AMIGA 500. JE GLANE DONC TOUT PLEINS D'INFOS SUR LES FORTH AMIGA. JE CHERCHE EN PARTICULIER DE LA DOC SUR L'IMPLEMENTATION DE MVP FORTH. QUI EST DISTRIBUE EN FREEWARE MAIS SANS DOC. SPECIALS GRETTINGS TO SECRETAIRE.

SECRETAIRE Du 24.02.89 A 09h06
CONCERNANT FORTH SUR AMIGA: NOUS NE SAVONS RIEN, MALHEUREUSEMENT. QUE CEUX QUI PEUVENT REPONDRE A TICK PRENNENT LA PAROLE, POUR MA PART, JE NE DIRAI RIEN, MEME SOUS LA TORTURE ET LES MENACES...

JLUC Du 26.02.89 A 10h58
REPONSE A UNE QUESTION DE A.JACCOMARD PARUE DANS JEDI N 33 (2 ANS). J'ESPERE QU'IL A EU SA REPONSE AVANT. ENFIN, SINON EN VOILA UNE: DANS JEDI N20 IL Y A UNE ERREUR DANS LE TEXTE. LA VALEUR ANNONCEE EST CELLE DE K ET NON DE 1/K. LES CALULS DE (1+4-N) C C1/2 ETC... DONNENT LA VALEUR K UNE VALEUR PLUS PRECISE EST KK=1,646760257 CE QUI DONNE 1/K=0,6072529351
POUR PLUS DE PRECISION IL FAUDRAIT AUSSI RECALCULER LES EPS(K) QUI SONT EGAUX A ATN(2PUIS-K) ET TRAVAILLER EN MULTIPRECISION. JE VAIS M'Y PENCHER ALLEZ TOUT VIENT A POINT A QUI SAIT ATTENDRE.

SECRETAIRE Du 07.03.89 A 12h09
NOUVEAUX SOFTS EN TELECHARGEMENT:
FORTH MATHS: Calculs multi-precision
FORTH SYSTEME: Memory MAP TF83
FORTH GESTION FICHIERS: Acces a TF83 resident depuis FOXBASE 2+. Le fichier FORTHFOX.TXT contient les

instructions necessaires a la metageneration d'une version modifiee de TEST.COM lequel est transforme ensuite en fichier .BIN par l'application utilisateur. Pour tout complement d'info ou de probleme, laisser message dans la BAL de LAMBERTPH.
EN DIVERS, MENU GENERAL:
KEYBIOS.COM et .DOC
ASK.COM et .DOC
CALC.COM et .DOC
CARDFILE.COM et .DOC; ce programme est particulierement interressant pour les etourdis qui recherchent leur agenda. Une fois CARDFILE charge, on peut appeler l'agenda par appui sur ALT-SHFTdroit et rajouter, rechercher, supprimer ou modifier une fiche.

FORTH7 Du 07.03.89 A 13h41
AMIGA (suite) $ 1000 REWARD !!! THE WORLD'S FASTEST PROGRAMMER CONTEST a ete gagne par le team Delta Research de San Rafael, California avec le JFORTH tournant sur AMIGA. Je n'en sais pas plus: si des AMIGA'ys savent ou trouver le JFORTH, merci d'en aviser JEDI ou ma bal. Mon A-2000 accepte bien Turbo-Forth mais c'est sur la carte passerelle PC evidemment...
euh FORUM JEDI CLEAR ???

FORTH7 Du 10.03.89 A 14h32
QUESTION: Pourquoi l'affichage de Turbo-Forth n'est-il pas accelere par le resident QWICKCRT (telechargeable sur ce serveur)?
REPONSE: (pas tous en meme temps!) Parce que Turbo-Forth travaille au niveau du DOS et non du BIOS dans un souci de compatibilite maximale. Alors pour tous les speedes voici une autre definition de (CONS) utilisant le BIOS et le moyen de la patcher dans le kernel de Turbo-Forth. Avec QWICKCRT installe, comparez un WORDS en mode FAST et SLOW...

```
\ Version BIOS de (CONS) primitive EMIT
HEX
CODE (BIOS-CONS) ( car -- )
     BX BX XOR    AX POP
     0E # AH MOV    10 INT
     NEXT   END-CODE   DECIMAL
: FAST  ( -- ) \ affichage par BIOS
  ATTRIBUTS OFF
  ['] (BIOS-CONS) 2+  ['] (CONS) ! ;
: SLOW  ( -- ) \ affichage par DOS
  ATTRIBUTS ON
  ['] (CONS) DUP 2+ SWAP ! ;
```

SECRETAIRE Du 14.03.89 A 10h49
MODIF DANS FORTHFOX.TXT (TELECHARGEMT) Il faut rajouter 1- en debut de definition de ENTER:

```
 : ENTER ( seg adr lg ---)
   1- >R 2DUP  ...etc...
```

sinon, la chaine FOX$ contient le caractere 0 a la fin.
Exemple d'utilisation depuis FOXBASE:

```
 STORE " DARK WORDS " TO TFSTRING
 CALL APPLIFOX WITH TFSTRING
```

et FOXBASE execute WORDS. Maintenant, si sous APPLIFOX vous faites:

```
 " BIEN RECU " FOX$ $! RETURN
```

dans un mot execute, vous pouvez recuperer la chaine dans TFSTRING:

```
  ? TFSTRING.
```

(Pas trop mal a la tete...?) A+

SECRETAIRE Du 14.03.89 A 14h40
ACCES A BTX; JEDI N'EST PAS LE SEUL SERveur proposant des softs en ASCII. Sur le reseau BTX, le serveur de WDR Computer Club donne aussi des softs ASCII.
Acces: 3622 ou 36224949 (selon PAV) puis *111076405# et 10 ... apres, vous avez interet a comprendre la langue de GOETHE...verstanden?
Egalement un autre serveur sur BTX, specialise PC:
Acces *208888# (PC-NET Telesoftware)
Allez-y, vous vous remettrez dans le bain de vos cours de langue que vous sechiez au lycee...

SECRETAIRE Du 16.03.89 A 12h29
APPELEUR AUTOMATIQUE (ET PAS REPONDEUR) ce n'est plus un gag; l'appeleur automatique existe. Pour transmettre un message a une personne si cette personne n'est joignable qu'aux heures ou vous meme ne l'etes pas:
3617 VOCALE
Ce service vous donne des bonus pour l'utilisation en 3614. Transmet tout message telephone a n'importe quelle heure, n'importe quelle date (tenez, preparez les voeux de bonne annee des mintenant, ou les anniversaires). Choix entre une voix d'homme, de femme; possibilite de joindre un air musical au message. Vous allez craquer...

SECRETAIRE Du 16.03.89 A 12h35
NE MOURREZ PAS IDIOT: L'annuaire TELEX est disponible par MINITEL en composant le 3616 SCRIP.
Mais quelqu'un connait-il le moyen d'expedier un telex depuis un minitel et sans etre abonne a un serveur "passerelle"?

MOSAIQUE Du 22.03.89 A 14h52
PROBLEMES DE TELECHARGEMENT ET AUTRE AVEC UN PC1 D'OLIVETTI. CELA SEMBLERAI VENIR DU PORT DE SORTIE SERIE QUI EST DU TYPE 8251 ET NON 8250 D'OU PROBLEME DE COMPATIBILITE RS232C. L'ADRESSE DU PORT SERIE DU PC1 EST:&0D0 AU LIEU DE &3F8 SUR LE 1ERE PORT D'UN PC ORDINAIRE. EN MODIFIANT CETTE ADRESSE CELA SERAIT-IL SUFFISANT POUR FAIRE FONCTIONNER KERMIT PAR EX. COMMENT RETROUVER CETTE ADRESSE? (SECTEUR ET No DE LIGNE) AVEC PCTOOL OU AUTRE.? MERCI DE VOTRE REPONSE.

SECRETAIRE Du 23.03.89 A 08h55
REPONSE A MOSAIQUE: Ah, voila le genre de colle coton. Une remarque tout d'abord, personelle, "OLIVETTI ne fait rien comme les autres". S'il est vrai que votre port serie est en 0D0, alors ca va etre galere pour adapter KERMIT en le bricolant avec PCTOOLS mais c'est possible:
- tout d'abord, copier KERMIT sur une disquette de travail
- lancer PCTOOLS
- demander la fonction SEARCH
- demander la recherche des octets F8 03 (format cellule 16 bits en memoire mode INTEL) dans le fichier KERMIT.EXE.
- si trouve, relancer la recherche. Si cette paire d'octet n'existe qu'en un ou deux, voire trois exemplaires, essayez de modifier ensuite, apres reexecution de SEARCH, cette paire d'octets en la remplacant par la paire 00 0D dans cet ordre.
Attention, apres chaque modification, sortez de PCTOOLS et lancez votre KERMIT modifie; pas la peine de massacrer des adresses qui ne seraient pas celles du port serie.
Comme on dirait a HARA-KIRI: merci professeur JEDI...

SECRETAIRE Du 23.03.89 A 09h06
META-GENERATEUR INTEL 8051: Avec l'aide de LEMAIGRE, j'ai mis au point le meta-generateur de code INTEL 8051 ecrit en TURBO-Forth et tournant sur PC.
En quelques lignes, le micro-controleur 8051 est une sorte de micro-processeur 8 bits, mais avec l'avantage de disposer dans un seul boitier d'un port serie, 4 ports // 8 bits, un timer, 128 ou 256 octets de RAM interne, 4 ou 8k EPROM/ROM interne.
Avec cette bestiole, vous evitez de construire un circuit classique style processeur, decodeur d'adresse...
En outre, le 8051 coute trois fois rien: environ 70 Fr piece. Il peut servir a toutes sortes d'applications: conversion // serie et inversement, controle d'automatismes, regulation, etc...

MOSAIQUE Du 28.03.89 A 16h34
BONJOUR M'SIEUR LE PROF JEDI. J'AI ESSAYER TON TRUC PLUSIEURS FOIS. J'AI RETOURNER LES ADRESSES DS TOUS LES SENS SANS AUCUN RESULTATS.. QUE FAIRE?

ESTHETE Du 30.03.89 A 11h09
TRES INTERESSE PAR FORTH 8051 IDEE:
8031 CMOS
1 RAM
1 EPROM
1 ADAPTATION MINITEL
= SYSTEME ECONOMIQUE D'INITIATION A FORTH ET AU PROCESS DISPONIBLE SOUS FORME DE EPROM+SCHEMA
PS: ALIMENTATION PAR MINITEL
SAUVEGARDE RAM CMOS PAR CAPA
VOS ECRAN CONSERVES SANS FLOPPY (GENIAL CMOS!).

SECRETAIRE Du 30.03.89 A 12h21
Concernant FORTH 8051: le noyau FORTH 8051 est en cours de creation et de test. Mais l'assembleur 8051 avec son generateur de code cible sera diffuse prochainement sur SAM*JEDI (annonce en temps opportun sur ce FORUM)
Apres etude des caracteristiques du 8051 et comparaison

avec celles du 8031, il y aurait compatibilite, donc possibilite d'adapter le noyau 8051 au systeme propose par ESTHETE dans le precedent message.
Une autre puce interessante a exploiter dans cette voie est le muP 68HC11 de MOTOROLA (quasi-compatible code 6809).

En fait, les micro-controleurs sont certainement la voie royale pour les applications de FORTH:
- compacite extreme (1 a 1,5x le code assembleur equivalent)
- vitesse d'execution (0,4 a 1x l'assembleur).
- tres grande rapidite de developpement
- facilite d'adaptation et modification du code a une nouvelle gamme de muP.

Le temps de developpement de l'assembleur 8051, en partant de rien, est estime a 10 jours, et ce par une seule personne. Le meme travail aurait demande un delai beaucoup plus long s'il avait ete realise en assembleur ou en C.

Maintenant, le noyau 8051 incluera un minimum de primitives en code machine, permettant ainsi une portabilite aisee vers des muP de future generation. Exemple, OVER est ecrit sous la forme:

```
 : OVER >R DUP R> SWAP ;
```

ce qui est plus court que sa version assembleur en 8051. Si une application a un temps d'execution critique, il sera toujours envisageable de redefinir les parties sensibles en les optimisant en code machine.

Attention, dans le cahier des charges du noyau FORTH 8051, il n'est pas prevu de creer un interpreteur FORTH; donc pas de WORDS ou LIST... On privilegie la metageneration de code sans en-tete.

Le systeme PC ou compatible reste dans cette application le systeme de developpement a partir duquel est controle et genere le code.

Le FORTH 8051 sera donc avant tout un produit a vocation industrielle pour la gestion et le controle de processus automatises. Son seul but: faire gagner du temps dans la phase developpement de programmes en remplacement de l'assembleur 8051 classique.

SECRETAIRE Du 30.03.89 A 12h44
MANUEL TURBO-FORTH 83-STD: Mille excuses a ceux qui l'attendent avec impatience. Il n'est pas encore acheve. Sa parution est estimee courant juin-juillet 89.
Si quelqu'un habitant PARIS ou la region parisienne peut nous preter une imprimante laser connectable a un PC (duree 8 a 10 jours) qu'il se fasse connaitre en laissant ses coordonnees dans la bal SECRETAIRE ou au SYSOP(Ndlr: problème réglé à ce jour).

SECRETAIRE Du 30.03.89 A 12h47
REPONSE A MOSAIQUE: C'est bien ce que je pensais, votre cas est desespere. Il y a plusieurs options possibles:
- casser les pieds a OLIVETTI pour qu'ils reglent votre probleme,
- revendre votre truc-chose et racheter un vrai compatible, en fer, avec plein de ports d'extension, capot style 2CV, clavier bien mecanique, extensible a souhait et sans surprise.
- trouver les sources de KERMIT (yark, yark, yark...), potasser le C (plus ou moins long selon votre etat d'avancement) et l'adapter a votre compatible exotique.
Vous seriez maso, c'est la derniere solution que je vous engage a suivre.

FORTH7 Du 30.03.89 A 20h07
On vous a deja montre comment lancer DEBUG depuis Turbo-Forth. Oui mais comment PILOTER un programme-fils interactif sans ligne de parametres?
Exemple: je veux ecrire 'DESAS DUP' et voir le code de DUP desassemble par DEBUG sans avoir rien d'autre a faire.
L'est malade c'uila: faudra bien qu'il frappe Uquelque:chose <CR> puis Q <CR> sous DEBUG, c'te bonne blague.
MEUH NON si j'ecris dans un fichier les commandes Useg:adr <CR> Q <CR> et si je dirige l'entree de DEBUG depuis ce fichier! Aucun langage ne peut faire ca aussi facilement que Turbo-Forth. En un ecran Minitel:

```
\ DESASSEMBLEUR via DEBUG.EXE
 INCLUDE PIP   \ redirections des E/S
 : DESAS  ( <mot> -- )
  BASE @ HEX
  " CON TO DBUG.MAC"   $EXECUTE
  CR ." U" DSEGMENT (U.) TYPE ." :"
  ' @ (U.) TYPE CR ." Q" CR     RESTORE
  " KBD FROM DBUG.MAC" $EXECUTE
  " PROGRAM DEBUG.EXE" $EXECUTE RESTORE
  " DEL DBUG.MAC"      $EXECUTE
  BASE ! ;
 EOF
```

Le fichier DBUG.MAC est temporaire: il contient des macros 'construites' pour DEBUG puis est detruit!

SECRETAIRE Du 31.03.89 A 12h27
CLE POUR ACCES CONFIDENTIEL A VOTRE SYSTEME: vous partez en conge et souhaitez restreindre l'acces a votre machine en votre absence. Une solutin consiste a poser une cle electronique. Mais comme ce genre de programme n'est pas courant (si? vous l'avez? quelle chance...) il va falloir le bricoler soi-meme. En TURBO-Forth, j'en ai fait une pour ainsi dire comme qui rigole:

```
 4 CONSTANT #LETTERS
 #LETTERS STRING A$
 #LETTERS STRING PASS$
 CREATE INPUT-CHAR 1 ALLOT
 : SIGNON ( ---)
  CR ." MOT DE PASSE:" ;
 : PASSWORD-INPUT ( ---)
 A$ DROP 0 A$ $!
 SIGNON  #LETTERS 0
 DO  KEY INPUT-CHAR C!  ." -"
    INPUT-CHAR 1 A$ APPEND$
 LOOP  A$ UPPER ;
VARIABLE #ACCESS
3 CONSTANT MAX-ACCESS
 : RESET-SYSTEM  ( ---)
 CR CR CR 79 177 REPLICATE
 CR 177 EMIT 18 SPACES BLINK
 ." BLOCAGE MACHINE, APPELEZ "
 ." L'OPERATEUR AGREE" ATTOFF
 17 SPACES 177 EMIT CR  79 177 REPLICATE
 BEGIN AGAIN ;
: ACCESSING ( ---)
 #ACCESS OFF
 BEGIN  #ACCESS 1+!  PASSWORD-INPUT
   A$ DROP PASS$ COMPARE 0= NOT
 WHILE
   #ACCESS @ MAX-ACCESS =
   IF   RESET-SYSTEM
   ELSE CR  BOLD 5 SPACES
     ." Mot de passe errone, "
     ." recommencez" ATTOFF
   THEN
 REPEAT
 BYE ;
```

Une fois ceci tape et sauvegarde dans un fichier nomme PASSWORD.FTH, tapez sous TURBO-Forth:

```
INCLUDE PASSWORD
" XXXX" PASS$ $! \ remplacer XXX par
' ACCESSING IS BOOT \ mot de passe sur
SAVE-SYSTEM C:\PASSWORD \ 4 lettres
BYE
```

Puis sous DOS, revenez a votre repertoi-re racine par CD \
Editez AUTOEXEC.BAT et rajoutez une ligne au debut de AUTOEXEC (de preference apres KEYBFR le cas echeant) avec la commande PASSWORD.

MAintenant, votre systeme est protege contre toute tentative d'utilisation frauduleuse. Attention, notez bien votre code d'acces.

FORTH7 Du 01.04.89 A 21h53
3h 47mn 26s pour le crible d'Erathos: le record de lenteur a ete battu par Turbo-Forth. Ce 'turbo' tournait sur un simulateur 8086 lui-meme en 'turbo' lequel installe sur un operating compatible MS-DOS en 'turbo' qui pour des raisons comprehensibles avait ete 'virtualise' sur le disque du PC...

POISSON D'AVRIL ?

Pas vraiment car songez a ceci: qui voudra d'un processeur Forth 50 Mips si on ne fournit pas avec l'emulation des 80xxx et un OS acceptant sans broncher MS-OS/2-UNIX?

Emulation des emulations; tout n'est qu'emulation! Turbo-Forth n'est-il pas un emulateur de machine Forth sur PC ? Alors ...

JLUC Du 03.04.89 A 22h01
SUITE DE L'IDEE DE FORTH7, CELLE QUI N'ETAIT PAS UN POISSON D'AVRIL. SUR DEBUG AUTOMATIQUE. CHEZ MOI C'ETAIT QUAND MEME UN POISSON D'AVRIL CAR CA NE FONCTIONNAIT PAS??? J'AI CHANGE LES CR PAR DES 13 EMIT ET CA MARCHE. AMICALEMENT.

SECRETAIRE Du 04.04.89 A 10h23
A TOUS, INDISPONIBILITE DU SECRETAIRE: Oui, je prends 10 jours de conge. En mon absence, soyez sages, pas de disputes... Si vous voulez me joindre, laissez un mot dans ma BAL ou au SYSOP.
Ah, a propos de JEDI no 50, je manque un peu de matiere. Alors, les forcats (devrai-je dire les FORTHats??...) de la plume, a l'ouvrage SVP, que ce ne soient pas toujours aux memes de trimer.
On acceptera de publier tout ce qui vous passera par la tete (sauf les eructations meurtrieres, style Adolf Khomeyny) et qui concerne tout langage.

FORTH7 Du 04.04.89 A 18h31
REDIRECTION DES ENTREES (suite) Vachement embetant cette histoire du DEBUG foireux chez JLUC. Ca veut dire en clair que sur certains systemes une entree CR doit etre limitee au seul <CR> (code 13) sans le <LF> (code 10). Bon, pour Forth pas de pb (JLUC a fait la modif) mais pour le pov' epsilon qui utilise un editeur? Pas facile d'en trouver un qui ne mette pas <CR><LF>!

J'ai l'impression (mais me goure-je?) que la version MS-DOS n'est pas en cause. Ce serait plutot au programme de comprendre une entree <LF>: celui qui a reecrit le DEBUG.EXE de ma 3.21 a pense a la redirection de son pgr!

SECRETAIRE Du 05.04.89 A 13h12
PPC PORTABLE AMSTRAD AVE DISQUE DUR: C'EST POSSIBLE MAINTENANT; CHEZ LOISITECH, FOURNITURE DU PPC PORTABLE AVEC DISQUE DUR 20MO INTEGRE. POSSIBILITE MONTAGE SUR VOTRE PPC.

CONTACT: STE LOISITECH
CENTRE COMMERCIAL TERMINAL 93
93100 MONTREUIL
TEL 1-48.59.12.57

JLUC Du 06.04.89 A 23h22
SUITE DE LA SUITE DE L'AUTRE SUITE MOI, J'AI UN EDITEUR QUI PERMET DE NE METTRE QUE DES <CR> A LA PLACE DES
<CR><LF> MEME QUE JE M'EN SERS DE TRAITEMENT DE TEXTE AVEC FORTH:
C'EST <PUB> NATHALIE
ET AVEC DEBUG POUR LES REDIRECTIONS CA MARCHE. A LA PROCHAINE

SECRETAIRE Du 08.04.89 A 09h41
IDEE DE PROGRAMME: MINITEL HD, (HAUTE DEFINITION). TRANSMETTRE UN GRAPHISME PAR MINITEL, OUI, C'EST POSSIBLE. VOICI COMMENT J'ENVISAGE LA CHOSE.

LA PLUPART D'ENTRE VOUS DISPOSE D'UNE LIAISON MINITEL PC.
- BASCULAGE DE LA LIAISON MODEM VERS PRISE SANS ECHO ECRAN,
- UN CODE 32 VAUT UN POINT NOIR,
- UN CODE 33 VAUT UN POINT BLANC,
- UN CODE SUPERIEUR A 33 SIGNIFIE QUE
LE PRECEDENT POINT EST A REPETER x-33 FOIS: EX: 35 SIGNIFIE QUE LE PRECEDENT POINT EST A REPETER 35-33, SOIT DEUX FOIS.
- EN FIN DE TRANSMISSION, RESTAURATION STATUT DES AIGUILLAGES INITIAUX DU MINITEL.

ON NE PEUT UTILISER LES CODES INFERIEURS A 32 SOUS PEINE DE PRENDRE LE RISQUE D'INJECTER UN CODE DE CONTROLE, STYLE DECONNECTION OU AUTRE.

LE PROGRAMME DEVRA TRANSMETTRE EN PREFIXE LA TAILLE X-Y ET LES COORDONNEES X-Y D'AFFICHAGE DU GRAPHISME.

LE PROGRAMME DEVRA PERMETTRE INDIFFEREMENT LA TRANSMISSION ENTRE SYSTEMES GRAPHIQUES DISPARATES: HERCULES<->CGA,CGA<->EGA, ETC...

JE PROPOSE, POUR NE PAS COMPLIQUER, DE TRANSMETTRE DANS UN PREMIER TEMPS DU GRAPHISME MONOCHROME. L'EMPLOI DE FENETRES EN LOCAL POUR LA SELECTION D'OPTIONS N'ES PAS EXCLU.

QUE CEUX QUI SOUHAITENT PLANCHER SUR CETTE IDEE OU SIMPLEMENT APPORTER DES SUGGESTIONS SE FASSENT CONNAITRE.

SI CE PROGRAMME VOIT LE JOUR, IL PERMETTRA DE TRANSMETTRE DES DESSINS, SCHEMAS, PHOTOS (PRISES PAR HANDI-SCANNER PAR EXEMPLE) ENTRE DEUX PC VIA MINITEL:
PC<->MNTL.....MNTL<->PC
ATCHAO, VELOMA, KENAVO, AUF WIEDERSEHEN, ET TUTTI QUANTI.

---

freedom33 Du 12.04.89 A 15h28
help: qui pourra me donner les adresses exactes du bios at (128ko et 256ko) afin de pouvoir les copier sur une disquette?????? Merci d'avance a vous tous.

DIDIER J Du 12.04.89 A 15h33
EXISTE T IL UN FORTH83 OU UN POLYFORTH SUR APPLE 2E ET SI OUI OU ET A QUEL PRIX? MERCI A TOUS

FLIPO Du 12.04.89 A 21h04
Je reviens des Journees FORTH d'Aix la Chapelle. Voici quelques nouvelles :
1) Nouveaux produits disponibles :
- Boitier a base RTX2000 a brancher sur RS232 d'un PC ou ATARI qui sert alors de terminal. Prix: 2000 DM (7000 FF) H.T.
Contact: Ulrich PAUL Erlenweg 18
D-8901 LEIFERSHOFEN
- Processeur RTX2000 nu 1777 DM T.T.C. pour les membres de la Forthgesellschaf (cotisation annuelle : 64 DM,incluant 4 numeros de Vierte Dimension)

2) Produits annonces :
-RTX4000 (Forth 32-bits) pour mi-90
-Par Zilog : Super-8 microcomputer a instructions de base en Forth.

3) Projet :des Journees FORTH d'Aix la Chapelle;
Realisation d'un FORTHscript (Postscript en Forth) a implanter par exemple sur un RTX2000. Utilisation sur une imprimante laser Atari par exemple.
Contact : Christoph KRINNINGER

4) FORTH 32-BITS sur Atari
La societe Bradley Forthware propose un Forth 32-bits pour Atari avec interface GEM, BIOS BDOS.

GAVROCHE Du 13.04.89 A 17h34
OU PUIS JE TROUVER MSKERMIT??? MERCI

JFW2 Du 14.04.89 A 12h10
SALUT A TOUS JE CHERCHE UN SCHEMA DE CONTROLEUR DE DISQUES DUR (10 ou 20 Mo) MERCI DE ME REPONDRE SUR BAL: JFW2. A+

SECRETAIRE Du 18.04.89 A 08h44
JE SUIS DE RETOUR: j'ai passe de bonnes vacances, un peu courtes, mais completement deconnecte de la micro-informatique.

Mon appel concernant d''eventuelles participations a l'elaboration d''un programme "MINITEL HAUTE DEFINITION" ne semble pas vous stimuler. Alors je repete ce que j'entends par MHD; il s'agit par l'intermediaire de deux MINITEL et deux PC de transmettre une image graphique (monochrome pour le moment):
PC---MINITEL ligne MINITEL---PC
En manipulant les aiguillages du MINITEL on peut demander l'envoi de code sur la seule prise peri-informatique du MINITEL ce qui preserve l'affichage texte de toute perturbation. Exemple:
- aiguillage modem > ecran minitel seul pour envoi de texte;
- aiguillage modem > prise pour envoi d'un graphisme haute definition
- en fin de graphisme, re-aiguillage modem > ecran minitel.
Pourquoi le choix du minitel<->PC? Parce qu'un minitel ne coute rien (il suffit de le demander a son ACTEL).

En codant de facon optimale le graphisme, une image haute definition peut etre transmise en un temps relativement court.

Bien entendu, ce codage doit permettre la restitution de l'image sur n'importe quel type de moniteur graphique (CGA, EGA, VGA, HERCULES, etc...) et meme de machines differentes (PC-minitel minitel-ATARI par exemple).

Ma premiere idee est de reconstituer d'abord l'image cote emetteur. Ensuite, on definit le cadre ou la fenetre graphique a transmettre pour eviter de transmettre toute l'image video. Puis le contenu de la fenetre est balaye point a point. Pour chaque point d'une couleur on comptera le nombre de points de couleur inchange.

En transmission, on enverra la couleur du point a afficher (noir ou blanc) puis le nombre de repetiions de ce point. Les restrictions: on ne peut transmettre un code inferieur a 32 a un minitel sans risquer de le perturber, voire d'interrompre la communication. Donc, toutes les valeurs transmises devront etre comprises dans l'intervalle 32-127. Certains codes inferieurs a 32 pourront quand meme etre exploites:

- code 13 pour signaler un retour a la ligne graphique suivante; evite de tracer une ligne vide
- code 12 pour repetition:

```
  33 12 55 signifiera
  |  |  |
  |  |  55-32=23 fois a repeter
  |  code repetition
  33-32=1; un point blanc a tracer
```

D'autres codes pourront etre reserves pour signaler au systeme recepteur que l'image est en video inverse. Ceci peut etre utile si le nombre de points colores est trop important; permet de diminuer le temps de transmission.

Les routines essentielles:

- en transmission: commande aiguillage minitel; test d'un point graphique; analyse des repetitions de points successifs de meme couleur (noir ou blanc);
- en reception: traitement des codes recus et tracage de l'image graphique.

Les methodes de productions d'image haute definition sont independants du programme de transmission reception. Elles peuvent etre generes de diverses maniere:

- logiciel DAO ou PAO
- image video digitalisee
- image calculee (fractal par ex.)
- image provenant d'accessoires d'instrumentation (convertisseur AD et logiciel type "OSCILLOSCOPE" par ex.)
- image par scanner ou handi-scanner.

A mon sens un tel logiciel serait assez simple a creer et permettrait de lancer un challenge accessibles aux programmeurs sur tout type de machine (PC, ATARI AMIGA, etc...) et a partir de differents langages (FORTH, PASCAL, C, assembleur).

Concretement, j'attends que vous deposiez dans ce forum toutes les idees, informations et bouts de programme que vous pouvez mettre a la disposition des autres connectants. Si chacun y met du sien, ce forum aura atteint son but, a savoir, reunir des competences autour d'un meme projet jusqu'a l'aboutissement de celui-ci.

Mettez vous au starting block, a vos marques, prets, partez!!!!

SECRETAIRE 18.04.89 12h22
REPONSE A FREEDOM33: copier le BIOS AT sur disquette. En principe, avec DEBUG vous avez ce qu'il faut pour copier votre BIOS sur disquette.

Taper DEBUG
puis sous DEBUG, taper:
 N BIOS.BIN
 R BX
affiche contenu de BX et ligne suivante
 : mettre valeur de BX modifiee
 R CX
affiche contenu de CX et ligne suivante
 : mettre valeur de CX modifiee
BX+CX=nombre d'octets a copier sur disque.
 W cs:off
ou cs et off sont les valeurs hexa du debut de votre ROM BIOS.
 Q pour quitter DEBUG.

Voila votre ROM BIOS sur disque. A vous de bricoler maintenant.

SECRETAIRE
21.04.89
16h13
AUX POSSESSEURS D'AMSTRAD: EN F83 LetP, j'aimerai faire essayer ce mot:

```
 : EID ( ---) [ HEX ]
   DUP C@ 01F AND 0 DO
     1+ DUP C@ 07F AND EMIT
   LOOP DROP ;
```

puis de rechercher la valeur du cfa de mot .ID en tapant
 ' .ID
Ensuite, par ' WORDS 50 DUMP, rechercher la position de la paire d'octets inversee du cfa de .ID dans le dump de WORDS. Notez cette adresse et faire:
 ' EID adr !
ou adr est l'adresse notee. Maintenant, par SEE WORDS, verifiez que .ID a bien ete remplace par EID dans WORDS. Executer WORDS et me dire si ca plante.

FF32 24.04.89 21h42
QUEL EST CE MOT DE LONGUEUR 0 SITUE APRES TRAVERSE DANS F83 AMSTRAD?

FF32 26.04.89 12h24
UTILISATEURS D'UNIX CONTACTEZ MOI SVP IL EST GRAND TEMPS QUE L'ON PORTE TF83; NOTEZ QUE NOUS SOMMES DEJA EN 1989 !

SECRETAIRE 26.04.89 13h26
PROJET F32: SI F32 DEVIENT UN VERITABLE MICRO, IL FAUDRAIT ENVISAGER SERIEUSEMENT LES MODALITES DU TRAITEMENT VIDEO. EN DEHORS DE CGA, EGA, VGA, HERCULES, UN AUTRE CHOIX EXISTE: D2-MAC.
AVANTAGE: CONVERSION BIT-MAP EN VIDEO TRES FACILE (PLUS EN TOUT CAS QUE BITMAP => VIDEO ANALOGIQUE RVB OU COMPOSITE). MAIS AVANT DE FAIRE CE CHOIX, IL FAUDRAIT CONNAITRE L'ETAT DE LA NORME D2-MAC. SI QUELQU'UN A DE LA DOC SUR CE SUJET, JE SUIS PRENEUR.

SECRETAIRE 29.04.89 13h27
PREMIERS MNEMONIQUES DE F32 DEJA DEFINIS
1) PILE A PILE:
 +, +C, -, -C, *,
 DUP, SWAP, OVER, DROP,
 =, 0=, >, <,
 0, -1, (flags booleens predefinis)
 OR, AND, NOT, XOR,
 >R, R>, R@,
2) MEM VERS PILE:
 CLIT, WLIT, DLIT, (8, 16 et 32 bits)
 C@, W@, D@,
3) PILE VERS MEM:
 C!, W!, D!,
4) BRANCHEMENTS:
 BRANCH, et ?BRANCH, (valeur relative 16 bits)
5) APPELS DE SOUS PROGRAMME:
 CALL,
 Nota: on combine sur 32 bits l'adresse et le type d'appel:
 bits 0 a 29, adresse relative ou absolue du sous programme;
 bits 30 et 31:
 01 = call relatif avant
 10 = call relatif arriere
 11 = call adresse absolue
Tous les cfa de mots seront codes en 32 bits selon cette methode en adressage relatif pour permettre de reloger le code.
Je prepare un assembleur F32 qui sera disponible des que possible en telechargement.

FORTH7 04.05.89 14h26
Liberte n'empeche pas rigueur: INDENTEZ vos lignes Forth pour une meilleure lisibilite!
 <DefMacro><Ctrl-CR>
 <CR><UP><Ctrl-Right><Down>
 <DefMacro>
2eme version permettant de couper une ligne sur un mot en maintenant l'indentation:
 <DefMacro><Ctrl-CR>
 <CR><CR><UP><UP>

<Ctrl-Right><Down><Del>
<DefMacro>

(Macros pour WordPerForth bien-sur!)

FORTH7 05.05.89 18h22
ET SI ON SUPPRIMAIT LE DUP DU FORTH? Les operateurs FORTH peuvent s'entendre comme non consommateurs de la pile ainsi le '+' pourrait-il faire (S n1,n2 n1,n2,n3) (n3=n1+n2) ou bien (S n1,n2 --- n1,n3) au lieu de (S n1,n2 --- n3) en Forth habituel avec comme regle de conserver les operandes afin de pouvoir les utiliser a nouveau. Pour eviter la saturation de la pile, on adopterait une pile circulaire (disons 2**8 cellules) avec perte automatique des plus vieilles cellules devenues tres probablement inutiles. Genial, non?
Eh bien, essayez! A genoux vous benirez ce foutu DUP avant peu...
Je parle bien-sur de l'utilisation d'une pile au plus bas niveau: on peut tres bien concevoir une programmation sans ces "horribles" DUP SWAP OVER DROP (voir variables locales ALEPH utilisant une pile accessoire).
Ceci dit, la recuperation d'operandes APRES l'operation est une idee pouvant se reveler pratique et plus rapide que le DUP avant. La touche LAST-X des calculateurs HP en RPN evite des ENTER (equivalents de DUP).
Malheureusement CODE LAST-X SP DEC SP DEC NEXT C; ne donne pas pour 3 4 + LAST-X * le 3 4 + 4 * attendu parce que l'interpreteur utilise la pile lui aussi! LAST-X marche bien (et vite) en compilation!

SECRETAIRE
08.05.89
10h51
CORDIC EN TURBO-FORTH:
J'AI RENCONTRE LAVARENNE QUI VIENT D'IMPLANTER LES TRAITEMENTS NUMERIQUES VIRGULE FLOTTANTE EN ALGORITHME DE CORDIC SOUS TURBO-FORTH. CEUX-CI SERONT DISPONIBLES SOUS PEU EN TELECHARGEMENT (AVIS SUR CE FORUM). SON PROGRAMME PERMET TOUS LES CALCULS SUR REELS EN PRECISION QUELCONQUE (12, 15, 20, 30 CHIIFRES APRES LA VIRGULE) DONC PRECIEUX POUR LES CALCULS ASTRONOMIQUES. POUR VOUS FAIRE BAVER UN PEU:
- OPERATEURS STANDARDS + - * / etc...
- OPERATEURS TRIGOS: SIN COS ASIN ACOS TAN ATAN SINH COSH etc...
- FONCTIONS HYPERB: SQR LOG LN EXP etc

ET MEME CALCULS SUR COMPLEXES, ROTATION VECTORIELLE, CONVERSION POLAIRE EN SCALAIRE etc...

DE PLUS, POUR CEUX QUI UTILISENT DEJA LE RTX2000 DE HARRIS, IL EXISTE CES MEMES ALGORITHMES DE CALCUL EN CORDIC, CREES PAR LAVARENNE, AU CATALOGUE DE MPE (MICROPROCESSOR ENGEENERING, HANLEY ROAD SOUTHHAMPTON, GB).

AVEC L'ARRIVEE DE CES NOUVEAUX PROGRAMMES DE CALCUL EN VIRGULE FLOTTANTE, TURBO-FORTH VIENT DE GAGNER SA PLACE SUR LE PODIUM DES VERSIONS FORTH PC LES MIEUX EQUIPES EN PROGRAMMES D'APPLICATION.

SECRETAIRE 12.05.89 16h24
L'equivalent de EMIT vers le port serie est:

```
  CODE >RS  ( c ---)
   SERIAL 5 + # DX MOV
   BEGIN  DX AL IN   20 # AL TEST
   0<> UNTIL
   SERIAL # DX MOV   AX POP   DX AL OUT  NEXT END-CODE
```

Et l'equivalent de KEY depuis le port serie est:

```
  CODE RS> ( --- c)
   SERIAL 5 + # DX MOV
   BEGIN  DX AL IN   01 # AL AND
     1 # AL TEST
   0<> UNTIL
   SERIAL # DX MOV   AX AX SUB   DX AL IN   1PUSH END-CODE
DECIMAL
```

SECRETAIRE 12.05.89 16h36
DECRYPTAGE D'IMAGES PCX: Les images PCX sont utilisees par WINDOW, PC-PAINTBRUSH etc... de MICROSOFT.
Maintenant, grace a turbo-Forth, on peut egalement les visualiser. Un programme nomme DECODPCX.FTH a ete installe en telechargement, accompagne de 4 fichiers d'images .PCX:
- VOLTAIRE.PCX pour les litteraires
- DYNAMO.PCX pour les survoltes
- SERPENT1.PCX boire ou conduire...
- GABY1.PCX quand sexe et soft se marient...

Ces images sont exploitables sur IBM PC ou compatible equipe d'une carte CGA et en mode video 640x200 (6 MODE sous TURBO-Forth).
Pour les moniteurs type HERCULES, VGA, EGA, vous pourrez adapter DECODPCX. (me communiquer les modifs, merci d'avance...).
Si des courageux veulent faire la meme chose en PASCAL ou en C, pas de probleme non plus.
Ce programme est un des modules du projet de minitel haute definition.
Bonnes cogitations (n'abusez pas de GABY1... si vous voulez, j'ai aussi GABY2 et GABY3 sous le coude...)

NOUVEAUX SOFTS EN TELECHARGEMENT:
FORTH:
Graphisme: DECODPCX.FTH et 4 fichiers d'extension .PCX; decodage d'images graphiques en CGA 640x200.
Maths: 8 fichiers pour faire vos calculs en virgule flottante selon les algorithmes de CORDIC. Selection de la precision a la compilation (12 a +100 chiffres apres la virgule. Dispose de sa propre pile de nombres flottants. Selon que l'on a choisi d'optimiser avec FP8086 ou FPFORTH, on dispose de routines optimisees en assembleur ou adaptables a d'autres FORTH que TURBO-Forth. Nota: le package CORDIC est en anglais.

SECRETAIRE 13.05.89 10h27
TRAVAIL PRO: J'AI POUR MISSION, DANS LE CADRE DE MES ACTIVITES PROFESSIONELLES, D'EQUIPER UN LABO DE GENIE CIVIL (MESURES ET TESTS) D'APPLICATIONS DE SAISIES ET TRAITEMENTS QUI DEVRONT S'EFFECTUER DEPUIS UN MINITEL.
J'ENVISAGE DE PROGRAMMER UNE PARTIE DU PROGRAMME EN FORTH, EN CE QUI CONCERNE LA PARTIE COMMUNICATION ET SAISIE.

POUR MENER A BIEN RAPIDEMENT CETTE PREMIERE ETAPE, JE RECHERCHE LES SPECIFICATIONS TECHNIQUES DU MINITEL 1B ET SURTOUT LES CODES DE CONTROLE EN ENTREE ET SORTIE. (STUM MINITEL 1B) (Ndlr: problème réglé à ce jour)
MON CAHIER DES CHARGES EST ASSEZ SIMPLE:
- SAISIE DE DONNES DEPUIS UN MINITEL COMMUNIQUANT AVEC UN PC.
- TRAITEMENT DES DONNEES SUR PC ET ENVOI DES RESULTATS VERS LE MINITEL
- REAIGUILLAGE DES DONNES DU MINITEL VERS LA PRISE MINITEL DISTANT AFIN DE PERMETTRE UNE IMPRESSION DES RESULTATS AU LABO.
- ENFIN, CONNEXION D'UNE BALANCE ELECTRONIQUE AU MINITEL POUR AUTOMATISER LA SAISIE DE MESURES (PESEE DE PONDERABLES EN PHASE DE DESSICATION).

LE CHOIX D'UNE TELLE SOLUTION EST IMPOSE PAR DES NECESSITES BUDGETAIRES ET AUSSI POUR NE PAS IMOBILISER DU MATERIEL BUREAUTIQUE UTILISE PONCTUELLEMENT.

LES DONNEES SAISIES DEVRONT ETRE STOCKEES DANS UN FICHIER dBASE NON INDEXE MAIS AVEC UN GRAND NOMBRE DE CHAMPS (+50 CHAMPS). CERTAINES DONNEES DEVRONT SUBIR DES CALCULS D'INTERPOLATION: TRACAGE DE COURBE A PARTIR DE POINTS DE MESURE.

SI VOUS ETES INTERRESSES PAR TOUTE OU PARTIE DES TRAVAUX DE CE PROJET (DANS LA LIMITE DE CE QUI EST DIFFUSABLE SANS ENFREINDRE LE SECRET PROFESSIONNEL), CONTACTEZ-MOI POUR ECHANGE DE BONS TUYAUX.
ATTENTION: DONNANT-DONNANT, CAR JE ME REFUSE A TRANSMETTRE LE FRUIT DE MES RECHECHES A DES PERSONNES QUI L'EXPLOITERAIENT SANS RENVOYER L'ASCENSEUR.

SECRETAIRE 13.05.89 10h53
NOUVEAUX SOFTS EN TELECHARGEMENT EN C, PASCAL ET DIVERS.

DANS LA RUBRIQUE DIVERS, LE PROGRAMME CARDFILE.COM EST MAITENANT ACCOMPAGNE DE SON SOURCE EN ASSEMBLEUR ET D'UNE VERSION BASIC.

LAMBERTPH 13.05.89 21h28
PETIT PROBLEME DE METAGENERATION (A PROPOS DES METHODES- CERTAINS ME TROUVERONS ENTETE) SOIT DANS PHENIX LES ENTREES:

```
FORWARD: EXIT
T: METHODS> [[ TRANSITION ]] DOES>
         [FORWARD] EXIT ;
```

ET DANS KERNEL UN UTILISATEUR DU MOT DE METAGENERATION :

```
 : INTEGER CREATE 0 , METHODS> @ ! ? ;
```

J'OBTIENS EN FAIT LA VERSION SUIVANTE VERIFIEE PAR SEE
: INTEGER CREATE 0 , DOES> EXIT ! ? ;
COMMENT SE FAIT-IL QUE @ AIT ETE MANGE PAR EXIT?

J'AI BIEN TROUVE UNE PARADE AVEC DANS KERNEL LA VERSION SUIVANTE:
: INTEGER CREATE 0 , METHODS> @ @ ! ? ;

HORS EN REGARDANT LES MOTS DE KERNEL QUI UTILISENT DOES> ON TROUVE SOUVENT UN @ QUI N'EST PAS DANS LA VERSION DE PHENIX

FORTH7 15.05.89 20h38
REPONSE A LAMBERTH (et esprits Forth) Le DOES> de TRANSITION utilise le mot le suivra pour resoudre le code executif des mots de 2eme generation. Pourquoi? Parce que vous aurez peut-etre besoin de metadefinir! Ajoutez

```
 FORWARD: <INTEGER>
 H: INTEGER TARGET-CREATE
         [FORWARD] <INTEGER> 0 , H;
(ca INTEGER de *** META *** )
(facultatif si pas de metadefinition)
: INTEGER CREATE 0 ,
         METHODS> <INTEGER> @ ! ? ;
(ca INTEGER de votre *** cible ***)
```

et vous pouvez alors METAcompiler qqch comme INTEGER MACHIN-CHOSE ds la cible! Vifs encouragements pour entetement!

SECRETAIRE 18.05.89 09h36
INITIALISATION PORT SERIE DU PC a 1200 bauds pour liaison PC<->MINITEL; le mot 1200-BAUDS diffuse il y a quelques jours doit etre modifie un peu; il faut egalement initialiser les parametres DTR et RTS:

```
HEX 03F8 CONSTANT SERIAL
CODE 1200-BAUDS  SERIAL 3 + # DX MOV
 80 # AL OUT  DX AL OUT  60 # AX MOV
 SERIAL # DX MOV  DX AL OUT  AH AL MOV
 DX INC  DX AL OUT  SERIAL 3 + # DX MOV
 1E # AL MOV  DX AL OUT
 SERIAL 4 + # DX MOV \ reg ctrl modem
 01 # AL MOV \ DTR set, RTS reset
 DX AL OUT
 NEXT END-CODE
```

a noter que l'initialisation du port serie peut aussi etre ecrite en FORTH en utilisant le mot PC!:

```
  HEX 03F8 CONSTANT SERIAL
: 1200-BAUDS ( ---)
 80 SERIAL 3 + PC!
 60 SERIAL PC!
 00 SERIAL 1+ PC!
 1E SERIAL 3 + PC! \ 7 bits, E, 1
 01 SERIAL 4 + PC! \ DTR set RTS reset
 ;
```

car cette routine n'etant utilisee que une seule fois, il n'est pas necessaire de l'optimiser en code machine.

CONCERNANT CONTROLE MINITEL 1B en mode 80 colonnes, les codes de controle IBM (sequences ESC, norme ANSI) sont applicables.

En mode 80 colonnes, on ne dispose pas de la couleur, mais seulement des attributs d'affichage suivants:

```
1B 5B 30 6D  pas d'attributs
1B 5B 31 6D  surintensite
1B 5B 34 6D  soulignement
1B 5B 35 6D  clignotement
1B 5B 37 6D  inversion video
```

Pour passer en 80 colonnes, il faut injecter au minitel les codes suivants: 1B 3A 32 7D et suivis de 0E si l'on veut le jeu ASCII francais avec caracteres accentues.

Si vous avez effectue la procedure de retournement du modem de votre MINITEL et qu'un correspondant dispose aussi d'un minitel 1B, cest le minitel du correspondant qui basculera en affichage 80 colonnes.

A partir de ce moment, le clavier etendu devient operationnel: touches flechees et envoi en bas de clavier, touches CTRL-car et touche ESC.

En esperant que ca vous titillera l'imagination... bye.

SECRETAIRE 20.05.89 9h16
MES ROUTINES DE COMMUNICATION MINITEL-PC AVANCENT BIEN. EN 80 COLONNES, ON PEUT GERER LES ATTRIBUTS DU MINITEL 1B AVEC LES MEMES SEQUENCES ANSI QU'UN IBM PC MONOCHROME (SEQUENCES ESC).

J'AI DEFINI LES EQUIVALENTS DE KEY, KEY? EMIT ." TYPE SANS PASSER PAR UNE REVECTORISATION SYSTEME. AINSI, UN MEME PROGRAMME FORTH EST UTILISABLE SUR PC OU MINITEL SELON QUE L'ON COMPILE AU PREALABLE LES ROUTINES D'INTERFACE OU NON.

MAINTENANT, JE PLANCHE SUR LES PROTOCOLES PERMETTANT D'IDENTIFIER LE TYPE DU TERMINAL DISTANT POUR AUTORISER LE BASCULAGE EN 80 COLONNES LE CAS ECHEANT.

LEMAIGRE 24.05.89 22h23
BONJOUR, PARTICULIEREMENT AU SECRETAIRE. J'AI FAIT UNE VERSION REORGANISEE DU FORTH.VOC POUR AVOIR LES INSTRUCTIONS CLASSEES SUIVANT LEUR DOMAINE D'APPLICATION. PAR EX:
GENERAL
MOUVEMENT DE PILE
ACCES MEMOIRE
ACCES DOS
INPUT OUTPUT
COMPILATION
CHAINES
ETC...
CELA DONNE UN ASPECT BEAUCOUP PLUS DOC ET BIEN PLUS UTILISABLE CA INTERESSE QUELQU'UN?
EN PLUS J'ESSAYE D'ECRIRE UNE MINI DOC DE DEMARRAGE POUR DES STAGIAIRES. SI CA VA BIEN POUR TOUTES LES INSTRUCTIONS QUI RESSEMBLENT AU FORTH79, JE PATAUGE UN PEU POUR LES EXEMPLES DE $EXECUTE, BIEN QUE JE PILLE SANS VERGOGNE JEDI. MAIS JE VOUDRAIS BIEN ARRIVER AU BOUT D'UN PETIT DOCUMENT DE DEMARRAGE DANS TF. JE CHERCHE NOTAMMENT COMMENT COMPILER LE NOM D'UN FICHIER POUR L'OUVRIR PAR OPEN SANS AVOIR BESOIN DE LE DONNER AU CLAVIER

SECRETAIRE 25.05.89 08h45
REPONSE PARTIELLE A LEMAIGRE: CONCERNANT FORTH.VOC; CE FICHIER NE DOIT PAS ETRE REORGANISE, CAR IL EST TRAITE PAR HELP. DONC L'RDRE DES MOTS DANS UN FICHIER DE TYPE .VOC IMPORTE PEU.

SI VOUS CREEZ UN NOUVEAU VOCABLAIRE, LOGO PAR EXEMPLE:

```
  VOCABULARY LOGO  LOGO DEFINTIONS
```

VOUS POUVEZ DOCUMENTER VOS DEFINITIONS DANS UN FCHIER LOGO.VOC. EX:

```
 : POUR  [COMPILE] : ; IMMEDIATE
 : FIN   [COMPILE] ; ; IMMEDIATE
 POUR ETOILE ASCII * EMIT FIN
```

MAINTENANT, POUR DOCUMENTER NOTRE EMBRYON DE LANGAGE LOGO, APPELER L'EDITEUR FORTH EN OUVRANT UN NOUVEAU FICHIER;

LE PREMIER MOT D'UNE LIGNE DOIT ETRE LE NOM DU MOT A DOCUMENTER:

```
POUR  Mot creant un nouveau mot dans
      le vocabulaire LOGO
```

LA FICHE D'AIDE D'UN MOT PEUT ETRE DE TAILLE QUELCONQUE (1 ligne minimum).

ON SEPARE LA FICHE D'AIDE DU MOT 'POUR' DE LA FICHE D'AIDE SUIVANTE PAR DEUX RETOURS CHARIOT; DEUX RETOUR CHARIOT= SEPARATEUR D'ENREGISTREMENT.

SAUVER VOTRE FICHIER AVEC LE NOM:
LOGO.VOC
PUIS REPASSEZ SOUS FORTH, (CELUI DANS LEQUEL POUR ET FIN SONT COMPILES), SELECTIONNEZ LE VOCABULAIRE LOGO:

```
 LOGO LEARN  FORTH  SAVE-SYSTEM LOGO
```

LEMAIGRE 25.05.89 20h25
OK. MERCI DES EXPLICATIONS. MAIS LE BUT EST UN DOCUMENT IMPRIME QUI PERMET UNE CONSULTATION PAR RUBRIQUE DE BESOINS DU GENRE: QU'EST CE QUE J'AI POUR FAIRE UNE MULTIPLICATION?
IMPOSSIBLE DE FAIRE AVALER A UN DEBUTANT QUE LE SEUL MOYEN EST DE FAIRE LE HELP DE TOUS LES MOTS AU HASARD, OU SE TAPER L'ORDRE ALPHABETIQUE POUR DECOUVRIR QU'IL EXISTE */MOD ET /MOD. LE HELP N'EST QU'UNE AIDE POUR UNE SYNTAXE. JE RESTE SUR MES POSITIONS! AMICALEMENT

FORTH7 26.05.89 14h45
SOUTIEN AU COLLEGUE LEMAIGRE! Votre initiative est

parfaitement dans le principe des fichiers de documentation *.VOC qui ont ete concus aussi bien comme fichiers-textes docs que comme index pour HELP.
C'est par simple paresse que nous avons adopte l'ordre alphabetique (pour ne rien oublier)!
Il suffit de repasser un fichier VOC plus "sympa" au LEARN pour re-indexer tous les HELP. De plus, il a ete prevu de pouvoir ajouter du texte "libre" (titres, paragraphes etc...): il suffit que le premier mot n'en soit pas un mot du dictionnaire a indexer.

Je signale cependant une limitation: un fichier .VOC ne doit pas depasser 64 Kilo-octets de texte pour que l'indexation des HELP soit correcte. Ceci est du au fait que le view-field servant de pointeur dans le fichier VOC est code sur 16 bits seulement. Cette limitation risque de poser un probleme assez serieux a qui voudrait documenter largement le vocabulaire FORTH qui est de loin le plus volumineux.

L'idee de fichiers .VOC structures est excellente: je suis pret a LEARN'er de nouvelles versions! (foreign versions likewise wanted!)

SECRETAIRE 26.05.89 18h19
JE CONSTATE QUE L'ON S'EXPRIME FERME AU SUJET DU BEBE TURBO-F83. DE MON COTE, J'AI PU TESTER LES FONCTIONS GRAPHIQUES DU FICHIER GRAPHIC.FTH EN MODE VGA:

```
  HEX 10 MODE  ou
      11 MODE  ou
      12 MODE
```

PERMET DE CHOISIR UNE COULEUR PARMIS PLUSIEURS:

```
      n GCOLOR
```

OU n VARIE ENTRE 0 ET LE NOMBRE DE COULEURS POSSIBLES DANS CES AUTRES MODES. EX: 12 MODE DONNE ACCES A 16 COULEURS AVEC UNE DEFINITION DE 640x480 PIXELS.

DANS LES MODES 4, 5 OU 6, LES CARACTERISTIQUES D'AFFICHAGE D'UNE CARTE VGA SONT SIMILAIRES A CELLES D'UNE CARTE CGA. COMME QUOI, PASSER PAR LE DOS EST UN GAGE DE PORTABILITE QUAND ON DEFINIT DES ROUTINES GRAPHIQUES.

MYCROFT 05.89 19h17
J'UTILISE AUSSI FORTH.VOC COMME FICHIER DOCUMENTAIRE IMPRIME 'A COTE'; COMME DEBUTANT JE PREFERE LA LISTE ALPHABETIQUE DE TOUS LES MOTS: CA VA PLUS VITE A CONSULTER POUR LIRE UN LISTING.
BIEN-SUR J'ATTENDS AUSSI LA VRAIE DOC PAPIER: C'EST POUR QUAND? A PROPOS DE FORTH.VOC, QUAND ON FAIT

```
   HELP MOT1 MOT2 MOT3 ...ETC...
```

TOUS LES MOTS APRES HELP SONT EXPLIQUES COMMENT FAIRE POUR QUE HELP S'ARRETE D'EXPLIQUER?

SECRETAIRE 26.05.89 21h01
CONCERNANT LA VRAIE DOC PAPIER: ELLE EST EN VOIE D'ACHEVEMENT. DES IMPERATIFS PERSONNELS ONT RETARDE SON ACHEVEMENT. D'AUTRE PART, J'AVAIS FAIT APPEL POUR LE PRET D'UNE IMPRIMANTE LASER (SUR REGION PARISIENNE) ET N'AI OBTENU AUCUNE REPONSE...

HEUREUSEMENT, MAINTENANT C'EST REGLE. DONC, JE VAIS POUVOIR TIRER LA MAQUETTE DU MANUEL DANS QUELQUES SEMAINES. REMERCIEMENTS AUX CORRECTEURS BENEVOLES

SECRETAIRE 26.05.89 21h05
CONCERNANT LE TRAITEMENT DE FICHIERS PAR MINITEL, J'AVANCE BIEN. CE PROGRAMME DESTINE A ETRE OPERATIONNEL DANS LE CADRE DE MES ACTIVITES PROFESSIONNELLES POURRAIT AUSSI INTERESSER DES ADHERENTS. ACTUELLEMENT, JE GERE L'ECRAN DU MINITEL 1B DISTANT EN 80 COLONNES AVEC SES ATTRIBUTS, POSITION CURSEUR. LORS DE LA SAISIE D'UNE CHAINE DE CARACTERE, ON PEUT FILTRER PRATIQUEMENT COMME AVEC UNE ZONE DE SAISIE dBASE.

LUNDI J'ATTAQUE LE TRAITEMENT NUMERIQUE SOUS LA FORME:

```
  x y AT " ###.## " PICTURE num$ GET
```

EN CAS DE FRAPPE DE CARACTERE NON NUMERIQUE 0..9 VIRGULE OU POINT, JE TRANSMET UN MESSAGE D'ERREUR SUR LA LIGNE 0 ET REMET LE CURSEUR EN ZONE DE SAISIE.

MES ROUTINES DISPOSENT DE PLUSIEURS FILTRES:

```
 NUMERIQUE 0..9 , .
 DATE      0..9 /
 ALPHA MAJ a->A..z->Z
```

LES DONNEES AINSI SAISIES POURRONT ETRE STOCKEES DANS UN FICHIER dBASE III/III+.

CES ROUTINES SERONT PARTICULIEREMENT UTILES A CEUX QUI VOUDRONT DECENTRALISER LA SAISIE OU CONSULTATION DE DONNEES A PARTIR D'UN MINITEL GERE COMME TERMINAL TELEINFORMMATIQUE.

SI VOUS MEME POSSEDEZ UNE CARTE CORTEX, JE VOUS PROPOSE DE M'AIDER A ADAPTER MES ROUTINES QUI NE GERENT POUR LE MOMENT QU'UN MINITEL RETOURNABLE CONNECTE SUR LE PORT SERIE DU MINITEL.

BUT DE LA MANOEUVRE, EN DEHORS DE MON APPLI PRO, PROUVER QUE FORTH PERMET L'ELABORATION D'UTILITAIRES AVANCES ET ADAPTABLES. A+

FORTH7 27.05.89 18h56
SOIGNEZ VOS COMMENTAIRES! On ne met jamais assez de commentaires dans ses programmes: combien de genies meconnus pour l'avoir oublie?

Usez donc largement des ( ) (S \ ou du EOF pour documenter vos oeuvres!

Avec l'adoption du fichier-texte source de nouveaux Forth (comme Turbo-Forth) proposent de placer des commentaires multilignes entre accolades alt-123 et alt-125 (non minitel malheureusement!) ou entre alt-125 et alt-123, les avis restent partages. Voici pour TF83 une adaptation de ce principe:

```
: (COMMENT)   ( -- state) \ patch RUN
  ASCII }  \ accolade fermante alt-125
  PARSE NIP SOURCE NIP <>
  IF   ['] STATE IS RUN  THEN STATE ;

: {  ( -- ) \ accolade ouvrante alt-123
  ['] (COMMENT) IS RUN
  [COMPILE] \            ; IMMEDIATE

{                   ( accolade ouvrante)
ceci est un commentaire: toutes les lignes apres { sont
ignorees par l'interpreteur jusqu'au symbole final
}                   ( accolade fermante)
```

LEMAIGRE 29.05.89 23h14
S O S
LA FONCTION

```
: INPUT
  PAD 20  2DUP 32 FILL EXPECT PAD 1- NUMBER? ;
: IN  / TEST DE INPUT
   CR ." INPUT"
   CR ." FLAG " .
   CR ." DN " D.
   CR ." SPAN" SPAN ?  PAD 1- 20 CR DUMP  ;
```

NE FONCTIONNE PAS A LA 1ere COMPIL, MAIS PARFAITEMENT ENSUITE, DES QUE L'ON A TAPE LE MOINDRE MOT FORTH.
SUR UN PROGRAMME AUTO BOOT, PAS MOYEN DE LUI FAIRE FAIRE QQ CHOSE. MAIS SI L'ON SORT ET QUE L'ON RELANCE LE PROG C'EST OK. LE DUMP DE PAD EST NORMAL LE NUMBER? NE FAIT PAS LA CONVERSION. UNE IDEE?

FORTH7 30.05.89 20h30
QUELQUES BOUEES DE SAUVETAGE . . .
1) NUMBER? utilise une chaine implicite s'appuyant sur un espace de separation.
2) EXPECT prend au clavier une chaine explicite

```
  PAD 1+ 20 EXPECT
```

3) on a neglige le passage ch.explicite

```
ch.implicite              SPAN @ PAD C!
l'espace d'appui        BL PAD COUNT + C!
avant conversion            PAD NUMBER? ;
```

4) le PAD evolue avec le dictionnaire et les compilations successives: il ne doit etre utilise que temporairement (PAD=bloc-note) pour un meme mot.
5) l'espace avant PAD (pad 1-) n'est pas libre mais reserve a l'interpreteur et aux conversions mumerique-alpha. autoboot? euh c'est surement le delco

---

SECRETAIRE 31.05.89 10h55
REPONSE A LEMAIGRE: J'AI CRU COMPRENDRE QU'IL VOULAIT FAIRE UN INPUT NUMERIQUE. MAIS POURQUOI CHERCHE-T-IL LA PETITE BETE ALORS QUE L'ON PEUT FAIRE PLUS SIMPLE:

```
 10 STRING TAMPON$
 TAMPON$ INPUT$ ( INPUT$ DEJA DEFINI DANS TURBO-FORTH)
```

PUIS SI VOUS ENTREZ UNE VALEUR NUMERIQUE LA LIGNE SUIVANTE:
TAMPON$ $EXECUTE
CONVERTIT VOTRE CONTENU DE TAMPON$ EN NOMBRE.
SI VOUS DEFINISSEZ 1989 CONSTANT ANNEE ET LORS D'UN TAMPON$ INPUT$ VOUS REPONDEZ ANNEE, L'EXPRESSION TAMPON$ $EXECUTE RENVOIE 1989.

SECRETAIRE 31.05.89 13h21
SCIENCE ET MEDIAS: DEPUIS LE SIECLE DES LUMIERES, LES PHILOSOPHES ETAIENT SOUVENT DOUBLES DE MATHEMATICIENS, CHIMISTES, MEDECINS... AUJOURD'HUI, DANS NOS MEDIAS, LA PART CONSACREE AUX SCIENCES DEVIENT CONGRUE. EN DEHORS DES BREVES DE M.CHEVALET PENDANT LES JOURNAUX TV, IL NE RESTE PRATIQUEMENT RIEN.

UNE EXCEPTION, M6 QUI UN DIMANCHE SUR DEUX DIFFUSE 'QUAND LA SCIENCE MENE L'ENQUETE'. DE QUALITE TRES INEGALE, SALUONS QUAND MEME CET EFFORT CULTUREL.

ET MAINTENANT, SUR LA SEPT (RESEAUX CAABLES ET SATELLITE), MERCREDI A 15h30: UN MAGAZINE 'SCIENCE' DONT LE CONTENU EST INCONNU A CE JOUR. SI VOUS AVEZ LA CHANCE DE LE VOIR OU L'ENREGISTRER, FAITES NOUS UNE BREVE.

TOUJOURS PAR SATELLITE, UNE FOIS PAR MOIS, SUR SKY CHANNEL, UNE EMISSION SCIENTIFIQUE DIFFUSEE A... 2h00 DU MATIN.

VOICI LE PAF EUROPEEN, CELUI QUI VEUT RELEVER LES DEFIS TECHNOLOGIQUES DE DEMAIN. DANS UNE SOCIETE EN CROISSANCE TECHNOLOGIQUES EXPONENTIELLE, ON VOUDRAIT NOUS LAISSER DANS L'IGNORANCE QUE CA NE M'ETONNERAI POINT.
C'ETAIT MON BILLET DE MAUVAISE HUMEUR...

LEMAIGRE 05.06.89 22h18
REPONSE A FORTH7: MERCI POUR VOS EXPLICATIONS SUR MON INPUT EFFECTIVEMENT DECALAGE D'UN ESPACE EN AVANT ET ESPACE D'APPUI RESOUT PB.

REPONSE A SECRETAIRE: MERCI POUR EXEMPLE $EXECUTE, MAIS PAS SECURISE POUR EMPLOI TOUS PUBLIC, J'AVAIS BESOIN D'UNE ENTREE NUM SURE ET GESTION DE ESCAPE PAR REDIRECTION DE TABLE CC. CA MARCHE SUPER. MAIS QU'EST CE QU'ON APPREND DANS LES IUT D'INFORM. SUR LA PROGRAMMATION STRUCTUREE. CA JE ME DEMANDE TOUJOURS! A QUANQ LA CORRECTION DE SAISIE DU SERVEUR POUR LES LIGNES TROP LONGUES? AMICALEMENT A TOUS

SECRETAIRE 08.06.89 08h43
MACROS DE SUBSTITUTION: en dBASE, il est possible de definir des macros de substitution avec le symbole &.
En C, les macros de substitution sont definies par #define. Ce mot permet aux habitues de PASCAL, par exemple, de reutiliser leurs READ et WRITE en le definissant en C par #define.

Et en FORTH? Croyez-vous que cela soit si difficile de creer un mot #DEFINE?

Si vous pariez, deconnectez-vous sur cette page et planchez sur cette colle. Par contre, si vous etes allergique a la cogitation solitaire, tapez SUITE...
...
Vous avez bien planche? Voici ma version de #DEFINE (10 mn pour la mettre au point):

```
: #DEFINE  ( str --- <mot>)
 CREATE
  DUP DUP C, HERE >R 0 C, DP +!  R> COUNT $! IMMEDIATE
 DOES>  1+ COUNT $EXECUTE ;
```

Ouf, c'etait dur... Pourquoi ne l'a-t-on pas pondu plus tot. A quoi cela peut-il servir? Pour votre education FORTHienne, tapez a nouveau SUITE (faut bosser un peu).
...
Notre quidam programmeur FORTH ayant immediatement pige tout l'interet de #DEFINE aura maintenant les moyens de compiler un programme 79-Standard en TURBO-Forth sans avoir a retoucher le source de son programme:

```
" THEN " #DEFINE ENDIF
" DARK " #DEFINE CLS
" 1- PICK " #DEFINE PICK   etc...
: TEST ( fl ---)
IF  CLS  ENDIF ; ( definition 79-Std)
```

La definition decompilee de TEST sera equivalente a:

```
: TEST IF DARK THEN ; \ def 83-Std
```

FORTH7 08.06.89 15h02
FILETS DE MACROS propose:

```
: #DEFINE ( <nom macro> string -- )
  CREATE  TUCK HERE PLACE
          1+ ALLOT IMMEDIATE
  DOES>   COUNT $EXECUTE ;
```

" Input Stream Macros"
" Compiler Macros"
Comment traduire ce type de macros que seul Turbo (j'allais dire Turbot) sait cuisiner si bien ? Nos nouveaux anglicistes (WelcomE !) ont-ils des idees?

LEMAIGRE 08.06.89 19h05
IL EXISTE UNE VERSION TIMER EN TELECHARGEMENT POUR AIDER A TESTER DES PRGS.
CA M'A DONNE UNE IDEE D'EXTENSION DE DEBUG. LUI FAIRE AFFICHER LE TEMPS D'EXECUTION DE CHAQUE MOT. PEUT ETRE UTILE QUAND ON EST ENCORE DANS LES MOTS DE BASES. CELA SERAIT UNE AIDE POUR DECOUVRIR POURQUOI VOTRE PROGRAMME EST SI LENT DES QUE L'ON EST A UN NIVEAU SUPERIEUR. SI CA VOUS TENTE,MOI JE NE M'Y ATTAQUERAI PAS. SLTS ■